NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

FEASIBILITY OF DEVELOPING ACADEMIC LABORATORIES USING A LOW-COST ROBOT

by

Antonio Valle

September 2009

Thesis Advisor:	Xiaoping Yun
Second Reader:	Alexander Julian

Approved for public release; distribution is unlimited

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** September 2009	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE Feasibility of Developing Academic Laboratories using a Low-cost Robot		**5. FUNDING NUMBERS**	
6. AUTHOR(S) Antonio Valle			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000		**8. PERFORMING ORGANIZATION REPORT NUMBER**	
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) Space and Naval Warfare Systems Center Pacific 53560 Hull Street San Diego, California 92152-5001		**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited		**12b. DISTRIBUTION CODE**	

13. ABSTRACT (maximum 200 words)

The objective of this research was to investigate the feasibility of developing new academic laboratories for an introductory robotics course at the Naval Postgraduate School (NPS) using low-cost commercially-available robots.

In particular, this research used a desktop computer with Fedora 8 Linux operating system, a wireless network and the Garcia robot from Acroname Incorporated. The Garcia robot is a wheeled robot that has many onboard devices, such as encoders, infrared sensors, and a laser range finder with the capability of further expansion.

The investigation of the feasibility of developing laboratories using a low-cost commercially-available robot yielded mixed results. The positive results were that a low-cost robot, the Garcia, was found to be a flexible and powerful academic tool. The Garcia robot allowed for the development and implementation of a collection of laboratories to ensure that basic robotic functions are understood. The drawbacks were that the Garcia robot was difficult to start due to the lack of proper documentation. Also, the selected host configuration limited the Garcia's performance because the configuration injected an initial latency of 15 to 20 seconds. The latency was noted when communicating with the robot and the laser simultaneously.

14. SUBJECT TERMS Robotics, Infrared, Encoders, Laser, Garcia			**15. NUMBER OF PAGES** 129
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

FEASIBILITY OF DEVELOPING ACADEMIC LABORATORIES USING A LOW-COST ROBOT

Antonio Valle
Lieutenant, United States Navy
B.S., United States Naval Academy, 2002

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 2009

Author: Antonio Valle

Approved by: Xiaoping Yun
Thesis Advisor

Alexander Julian
Second Reader

Jeffrey B. Knorr
Chairman, Department of Electrical and
Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The objective of this research was to investigate the feasibility of developing new academic laboratories for an introductory robotics course at the Naval Postgraduate School (NPS) using low-cost commercially-available robots.

In particular, this research used a desktop computer with Fedora 8 Linux operating system, a wireless network and the Garcia robot from Acroname Incorporated. The Garcia robot is a wheeled robot that has many onboard devices, such as encoders, infrared sensors, and a laser range finder with the capability of further expansion.

The investigation of the feasibility of developing laboratories using a low-cost commercially-available robot yielded mixed results. The positive results were that a low-cost robot, the Garcia, was found to be a flexible and powerful academic tool. The Garcia robot allowed for the development and implementation of a collection of laboratories to ensure that basic robotic functions are understood. The drawbacks were that the Garcia robot was difficult to start due to the lack of proper documentation. Also, the selected host configuration limited the Garcia's performance because the configuration injected an initial latency of 15 to 20 seconds. The latency was noted when communicating with the robot and the laser simultaneously.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF SYMBOLS, ACRONYMS AND ABBREVIATIONS

A/D	Analog to Digital
DCHP	Dynamic Host Configuration Protocol
DOF	Degrees Of Freedom
ERA-MOBI	Erratic Mobile Robot
FFUART	Full Function Universal Asynchronous Receiver/Transmitter
GHz	Gigahertz
GP	General Purpose
I/O	Input/Output
IP	Internet Protocol
I2C	Inter-Integrated Circuit
LEDs	Light Emitting Diodes
mAh	Milliampere-Hours
Mbps	Megabytes per Second
MCWL	Marine Corps Warfighting Laboratory
MHz	Megahertz
NiMH	Nickel Metal-Hydride
NPS	Naval Postgraduate School
PC	Personal Computer
PDA	Personal Digital Assitant
PID	Proportional Integral Derivation
PWD	Pulse Width Modulation

P3DX	Pioneer 3DX
RF	Radio frequency
scp	Secure Copy
SLAM	Simultaneous Localization and Mapping
ssh	Secure Shell
TCP	Transmission Control Protocol
USB	Universal Serial Bus
UAV	Unmanned Aerial Vehicle
u-boot	Universal Bootloader
UGV	Unmanned Ground Vehicles

EXECUTIVE SUMMARY

The military has taken note of the endless possibilities and reaches of robots. For years, the military has implemented robots in the battlefield. One prominent example is the Unmanned Aerial Vehicle (UAV), which removes a pilot from possible danger, as well as increases dwell time and saves money because it is less costly to employ and maintain. Other lesser known examples are the Unmanned Ground Vehicles (UGV), many of which have recently been deployed to Iraq. One UGV that stands out is the Dragon Runner, which is a product of collaboration between Carnegie Mellon Robotics Institute and the United States Marine Corps Warfighting Laboratory (MCWL). The Dragon Runner can gather valuable intelligence in an urban setting through its many onboard sensors while removing Marines from harm's way. The Dragon Runner's abilities and utility motivated this research to develop a platform for understanding basic principles of robotics.

An initial objective for this research was to find a low-cost commercially available robot. More importantly, the main objective of this research was to investigate the feasibility of developing new academic laboratories for an introductory robotics course at Naval Postgraduate School (NPS).

The investigation of the feasibility of developing laboratories using a low-cost commercially-available robot yielded mixed results. The positive results were that a low-cost robot, the Garcia, was found to be a flexible and

powerful academic tool. The Garcia robot allowed for the development and implementation of the following laboratories:

Laboratory One: Simple Maneuvers.

Laboratory Two: Getting Acquainted with the Garcia's Infrared Sensors.

Laboratory Three: Obstacle Avoidance.

Laboratory Four: Corridor Travel Using IR Sensors.

Laboratory Five: Stationary Mapping Using the Laser.

Laboratory Six: Mobile Mapping Using the Laser.

The drawbacks were that the Garcia robot was difficult to get started because there was not a complete set of instructions for an initial set-up. Additionally, the company was in transition between older and newer software and allowed the use of their pre-release alpha software, which meant that some issues were not completely addressed. Some of the code provided by the company had errors, but the technical support engineers were quick to address the errors. Once past the initial issues, the Garcia robot was fairly easy to use. Unfortunately, the host configuration selected for this research limited the Garcia robot's performance. Due to the configuration, an initial latency of 15 to 20 seconds was observed, especially when communicating with the robot and the laser.

ACKNOWLEDGMENTS

I would like to express my gratitude to the Naval Postgraduate School for the opportunity to pursue higher education.

Thank you to Space and Naval Warfare Systems Center Pacific for believing in my ability. Additionally, thank you for funding my research.

Thank you to technical support engineers at Acroname Incorporated for fielding all of my questions.

Thank you to Professor Yun for reigniting my interest in robotics, as well as for your leadership and guidance. Working with you was a real pleasure because you allowed me to experiment outside of my comfort zone in order to reach a higher potential.

A special thanks to laboratory engineer James Calusdian for your ever willingness to help and words of encouragement. Without your steadfast encouragement, the frustration in the early phases of research would have proved too much. You are truly a role model for work ethic and persistence.

Thank you to Derek Dye for assisting me with the C++ programming. Without your assistance, my thesis would have never gotten off the ground. Additionally, I would never be able to repay the amount of time you spent with me explaining the basics.

Thank you to my parents, Hernan and Sigda, for allowing me the chance to have the American dream. Your endless guidance, mentorship and can-do mentality has

provided me a solid foundation for life. More importantly, thank you for keeping me grounded and instilling the thought that with hard work anything can be done.

Finally, and most importantly, thank you to my beautiful wife, Melanie, and my son, Nason. You both make my life complete. After a hard day at work your smiling faces make it all worth it. I hope to never let you down.

I. INTRODUCTION

A. BACKGROUND

As technology continues to evolve, it allows for the development and implementation of robots into our surroundings. Robots can now be found in everyday life, taking over tedious and dangerous jobs from their human counterparts. The ranges of robots are endless, whether it is being used in a mail room to maximize accuracy and efficiency or as an exploratory nomad in a disaster area to minimize human exposure to dangers.

The military has taken note of the endless possibilities and reaches of robots. For years, the military has implemented robots in the battlefield. One prominent example is the Unmanned Aerial Vehicle (UAV), which removes a pilot from possible danger, as well as increases dwell time and saves money because it is less costly to employ and maintain. Other lesser known examples are the Unmanned Ground Vehicles (UGV), many of which have recently been deployed to Iraq. One UGV that stands out is the Dragon Runner, which is a product of collaboration between Carnegie Mellon Robotics Institute and the United States Marine Corps Warfighting Laboratory (MCWL). The Dragon Runner is described in [1] by a project manager as the following:

> The Dragon Runner is a small, lightweight, portable mobile reconnaissance/scout robot (or "bot"). At 15.5 inches long, 11.25 inches wide and 5 inches high, it is a tough low-lying/low-observable ground sensor. Dragon Runner is outfitted with a small video camera, an audio

microphone, infrared illuminators (for night operations) and infrared sensors (for obstacle avoidance).

The Dragon Runner can gather valuable intelligence in an urban setting through its many onboard sensors while removing Marines from harm's way. The Dragon Runner's abilities and utility motivated this research to develop a platform for understanding basic principles of robotics.

B. RESEARCH OBJECTIVES

An initial objective was to find a low-cost commercially available robot, although, the main objective of this research was to investigate the feasibility of developing new academic laboratories for an introductory robotics course at Naval Postgraduate School (NPS) using a low-cost commercially available robot.

C. APPROACH

This research used existing laboratory conditions to determine the feasibility of developing academic laboratories on a low-cost commercially-available robot. Specifically, this research used a desktop computer with Fedora 8 Linux operating system, a wireless network and a low-cost robot. The robot must be a wheeled robot that has many onboard sensors such as encoders, infrared and a laser range finder with the capability of further expansion. Furthermore, the robot needs to be programmed in C++, which is a common programming language among engineers.

D. THESIS ORGANIZATION

- Chapter I introduced research goals and presents the organization of the thesis.
- Chapter II presents a brief comparison between three low-cost robots that have similar capabilities as the Dragon Runner.
- Chapter III provides an overview of the Garcia Robot’s internal hardware, external features and software interaction.
- Chapter IV introduces movement laboratories.
- Chapter V introduces infrared laboratories.
- Chapter VI introduces laser laboratories.
- Chapter VII addresses conclusions and future research opportunities.
- Appendix A provides initial set-up/out-of-box procedures for the Garcia robot.
- Appendix B is Laboratory Zero: Getting Acquainted with the Garcia robot’s Directories and Linux Commands.
- Appendix C is Laboratory One Code.
- Appendix D is Laboratory Two Code.
- Appendix E is Laboratory Three Code.
- Appendix F is Laboratory Four Code.
- Appendix G is Laboratory Five Code.

- Appendix H is MATLAB Code associated with Laboratory Five.
- Appendix I is Laboratory Six Code
- Appendix J is MATLAB Code associated with Laboratory Six.

E. CHAPTER SUMMARY

This chapter provided a brief background and research goals. The chapter concluded with the organization of this thesis.

Chapter II presents a brief comparison among three low-cost robots.

II. A COMPARISON BETWEEN THREE LOW-COST ROBOTS TO DETERMINE RESEARCH SUITABILITY

A. DESIRED SPECIFICATIONS AND ONBOARD INSTRUMENTATION

Reading about and viewing videos of the Dragon Runner UGV sparked interest and thirst for knowledge of robots and their operation. Replicating the capabilities and performance of the Dragon Runner is beyond the scope of this research. Rather, this research intends to find a low-cost, commercially-available robot that can be used, in an academic setting, to provide an understanding of basic robotic functions that make a complex robot perform. This research intends to find and use a robot with similar specifications and onboard instrumentation as that of the Dragon Runner. Again, the dimensions of the Dragon Runner are the following: Length of 39.4 cm, width of 28.6 cm and height of 12.7 cm. This research imposed boundary conditions on the physical dimensions for a desired platform in order to expose students to the virtues and challenges of working with smaller platforms; therefore, the desired platform cannot exceed the Dragon Runner's physical dimensions by more than 50% in the length and width. In addition, the height cannot exceed more than 75%. Furthermore, the desired platform needs to include two types of range sensors to conduct obstacle detection and avoidance. In addition, the robot needs to have the capability for further expansion and wireless interactions. The following robots meet the desired demands and will be discussed in the next sections: Erratic (ERA-MOBI) by Verdiri Design, Pioneer 3DX (P3DX) by Mobile Robots Incorporated, and Garcia Robot by Acroname

Incorporated. Note that the prices listed in the following sections do not include tax or shipping fees.

B. ERRATIC (ERA-MOBI) BY VERDIRI DESIGN

The ERA-MOBI, seen in Figure 1, is a compact, powerful and industrial-strength mobile robot that was designed for scholastic research and application. This robot is within the desired physical specifications, measuring 40 cm in length, 41 cm in width and 15 cm in height.

Figure 1. Base Model of the Erratic Mobile Robot [From 2].

Furthermore, the ERA-MOBI can be configured with two types of range sensors, the LV-MaxSonar-EZ1 high-performance sonar range finder from MaxBotics, with a maximum range of 6 meters, and the Hokuyo URG laser range finder, with a maximum range of 4 meters, which can be used to conduct obstacle detection and avoidance. Additionally, this robot has the capability for further expansion to include an integrated personal computer (PC), pan tilt unit and a stereo camera. Also, the ERA-MOBI has the flexibility to

interact wirelessly with a host desktop computer or carry the host computer onboard. The ERA-MOBI has many configurations, but for this comparison, the model includes the following: the base ERA-MOBI model with the additional sonar rings and Hokuyo URG laser range finder. This robot is priced at $5,850.00, based on a price list provided by the manufacturer, and fulfills the desired specifications.

C. PIONEER 3DX (P3DX)

The P3DX, seen in Figure 2, is a robust mobile platform designed for research. This robot is within the desired physical specifications, measuring 44 cm in length, 38 cm in width and 22 cm in height.

Figure 2. P3DX with Two-Dimensional Laser Mapping and Navigation and Complete Tracking, Vision and Surveillance Systems [From 3].

This platform can carry a significant payload of up to 23 kg. Furthermore, the P3DX can be configured to carry two types of range sensors, a range finding sonar and a laser range finder. Also, this robot has the capability for

further expansion to include an arm with seven degrees of freedom (DOF), a color camera, stereo range finder camera, color tracking, bumpers and a gripper. For this comparison the P3DX model includes the following: the base P3DX model with the additional sonar rings and laser range finder. This robot is priced at $6,190.00, based on a price list provided by the manufacturer, and fulfills the desired specifications.

D. GARCIA ROBOT BY ACRONAME INCORPORATED

The Garcia robot, seen in Figure 3, is a mobile robot designed for research and end-user applications. This robot is within the desired physical specifications, measuring 28 cm in length, 20 cm in width and 12 cm in height.

Figure 3. Base Model of the Garcia Robot [From 4].

This flexible platform can be configured to carry two types of range sensors, an infrared sensor and the Hokuyo URG laser range finder with a maximum range of 4 meters, which can be used to conduct obstacle detection and

avoidance. Furthermore, this robot has the capability for further expansion to include text-to-speech, pan tilt unit and a camera. For this comparison the Garcia model includes the following: the base Garcia model with an additional Gumstix processor and laser range finder. This robot is priced at $4,548.00, based on a price list provided by the manufacturer's Web site, and fulfills the desired specifications.

E. A SIDE-BY-SIDE COMPARISON

The three robots introduced in the previous sections meet the desired criteria of commercial-availability, low cost, size and flexibility. All robots have been built for academic research; therefore, it is necessary to compare the robots by additional parameters that this research deems important, such as sensors, drive type, platform size, cost and weight.

Feature	ERA-MOBI	P3DX	GARCIA
Base Platform Size: L x W x H (cm)	40 x 41 x 15	44 x 38 x 22	28 x 20 x 12
Drive Type	Differential	Differential	Differential
Maximum Speed (m/sec)	2.0	1.2	1.0
Motors	DC reversible	DC reversible	DC reversible
Encoder (counts/revolution)	500	500	3648
Power (V)	12.0	12.0	7.2
Maximum Payload (Kg)	20.0	23.0	2.0
Total Weight including Batteries (Kg)	12.0	14.0	~ 2.0
Infrared Sensors	X		X
Sonar Sensors	X	X	
Laser Range Finder	X	X	X
Sensor Expansion Capabilities	X	X	X
Cost (as of August 2009)	$5,850.00	$6,190.00	$4,548.00

Table 1. Comparison of the Three Robots [After 5].

The three robots share some characteristics, but after reviewing the differences listed in Table 1, it was decided to use the Garcia robot from Acroname Incorporated, even as the ERA-MOBI and P3DX outperform the Garcia maximum speed

and load capacity. The higher encoder count per revolution provides the ability to perform more accurate maneuvers. Also of importance, the Garcia robot’s price is lower than that of the ERA-MOBI and P3DX. In addition, the Garcia offers the same degree of expansion for further research opportunities as the other two robots.

F. CHAPTER SUMMARY

This chapter provided a brief introduction and comparison of the ERA-MOBI, P3DX and Garcia that were considered for this research. Additionally, this chapter discussed the reasons why the Garcia robot was best fit for this research.

Chapter III presents an overview of the Garcia robot’s internal hardware, external features and software interactions.

III. SYSTEM OVERVIEW

A. GARCIA ROBOT'S INTERNAL HARDWARE

The major components of the Garcia robot's internal hardware, seen in Figure 4, are discussed in the following sections.

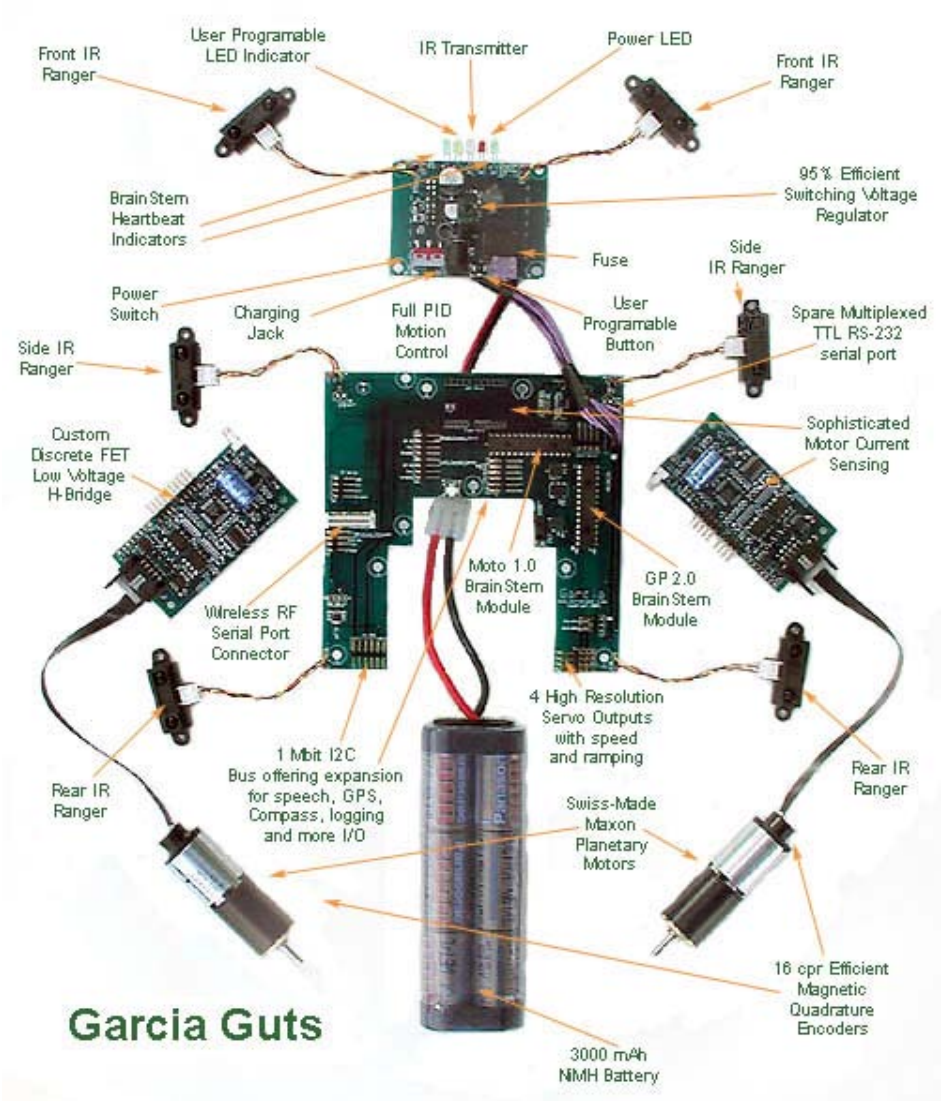


Figure 4. The Internal Components of the Garcia Robot [From 6].

1. General Purpose (GP) 2.0 Brainstem Module

The GP 2.0 is a controller that can handle a myriad of tasks simultaneously such as run stand-alone code, establish communication with a host computer and monitor reflexive actions. Additionally, the GP 2.0 Brainstem has the ability to perform voltage regulation and monitor for low-power voltage. Once low-power voltage is detected, the module forces the Garcia robot to shut down. Additionally, the GP 2.0 affords the following interfaces: five 10-bit A/D inputs, five digital I/O pins, 1 Mbps I2C bus, serial, and four high resolution pulse width modulation servo outputs. Also, GP 2.0 has the ability to execute 9000 instructions per second, making this module both powerful and flexible enough to handle any educational scenario.

2. Brainstem Moto 1.0 Module

The Moto 1.0 module offers motion control. It has two channels for motion control using Pulse Width Modulation (PWM) or Proportional Integral Derivation (PID). Furthermore, this module handles different types of feedback to include both digital and analog. The Moto 1.0 module, coupled with a three amperes low-voltage H-bridge and a 6-volt precision gear motor, provides smooth motion control of the Garcia robot.

3. Three Amperes Low Voltage H-Bridge

As the name states, this custom low-voltage H-bridge maintains a constant three amperes capacity. The H-bridge is connected directly to both the Moto 1.0 module and the precision gear motor, allowing for precision motion sensing.

Furthermore, this H-bridge allows for high-frequency switching capacity and current sensing, as well as a connection to the Maxon motors.

4. Maxon A-Max Motor

This Swiss-made motor provides a 19:1 gear-train reduction and is coupled with a 16 count per revolution quadrature encoder. Additionally, the Maxon A-Max motor can function with an input range of 4 to 8 volts. Also, the motor produces minimal noise during operation.

5. Infrared Sensors

The Garcia robot comes equipped with eight infrared sensors that can detect distances from 4 to 18 inches. The sensors are used for obstacle avoidance and wall-hugging functions, as well as monitors for drop-offs on the floor.

6. Gumstix Verdex Pro Motherboard

The Gumstix Verdex motherboard, seen in Figure 5, is comprised of a Marvell 600 MHz PXA270 Intel processor and many interfaces. The processor is physically small but delivers a substantial degree of computing power and flexibility. The Gumstix adds flexibility to the Garcia robot by providing wireless 802.11 networking capabilities and wireless Ethernet connectivity. Additionally, the motherboard is USB functional, which affords the addition of USB devices, such as a laser range finder and web cameras, to name a few. Furthermore, the motherboard provides a physical mount for the wireless antenna. Also of importance, the Gumstix communicates with the Brainstem modules via a serial link.

Figure 5. Gumstix Verdex Pro Package [From 7].

7. Batteries

The battery pack is composed of six Nickel Metal-Hydride (NiMH) batteries. The battery pack can produce an output of 7 to 9 volts. Additionally, the battery pack is rated for 4200 milliampere hours (mAh), which translates into several hours of continuous operation.

B. GARCIA ROBOT'S EXTERNAL FEATURES

In this section the Garcia robot's external features are discussed, such as Infrared and Laser range sensor location, and the head.

1. Infrared Sensor location

The Garcia robot has a total of eight Infrared sensors. The position of the sensors can be seen in Figure 6 and Figure 7. Six of the eight sensors provide the Garcia robot

a 360-degree view of its surroundings, and the remaining sensors, seen in Figure 8, are downward-looking to keep the Garcia robot from falling off a ledge.

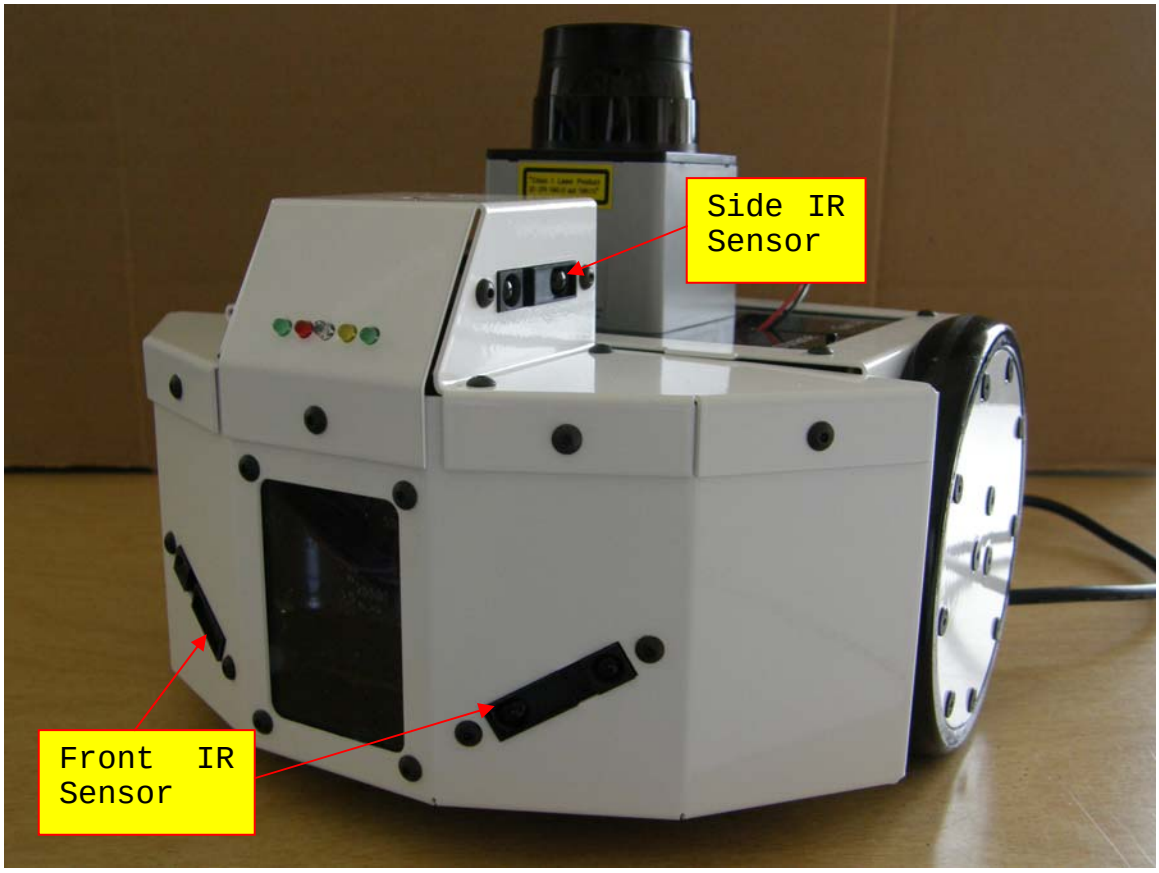


Figure 6. Front and Side Infrared Sensor Location on the Garcia Robot.

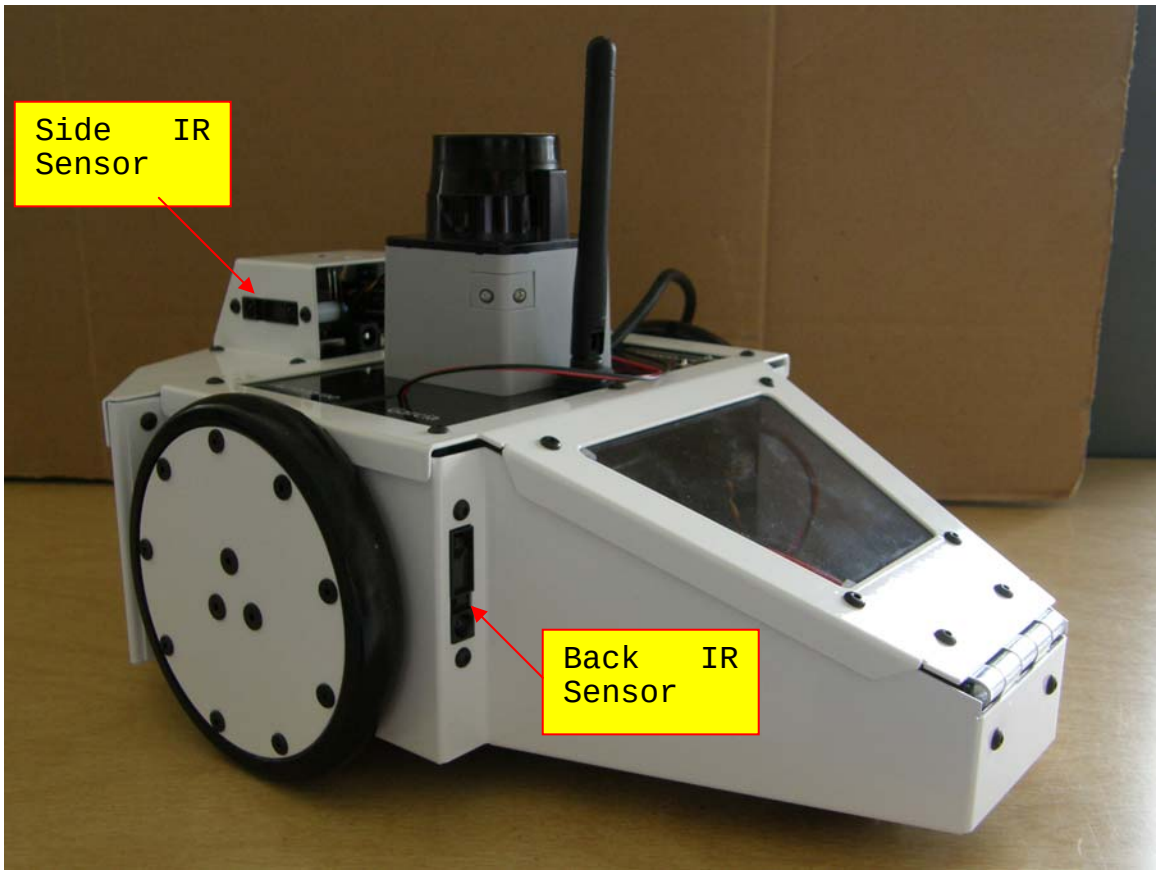


Figure 7. Back and Side Infrared Sensor Location on the Garcia Robot.

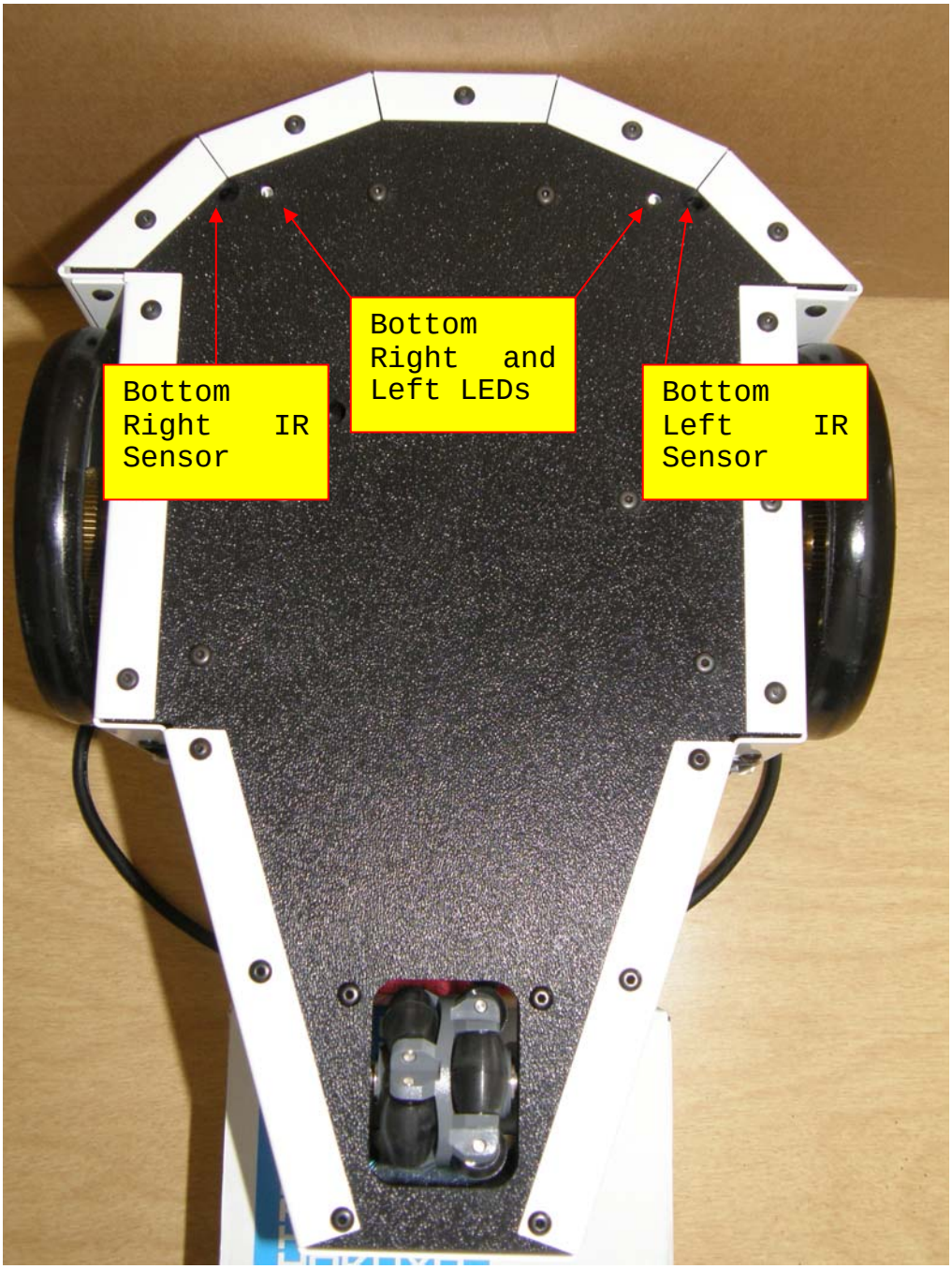


Figure 8. Infrared Sensor Location under the Garcia Robot.

2. Hokuyo URG-04 LX Laser Range Finder

The Hokuyo laser range finder is located on the top of the Garcia robot, seen in Figure 9. The URG-04 LX is an accurate laser and its specifications are listed in Table 2.

Specifications	
Detection Range (meters)	0.02 to 4
Scan Angle (Degrees)	240
Scan Time (msec)	100
Angular Resolution (Degrees)	0.36

Table 2. Hokuyo URG-04 LX Laser Range Finder Specifications [After 8].

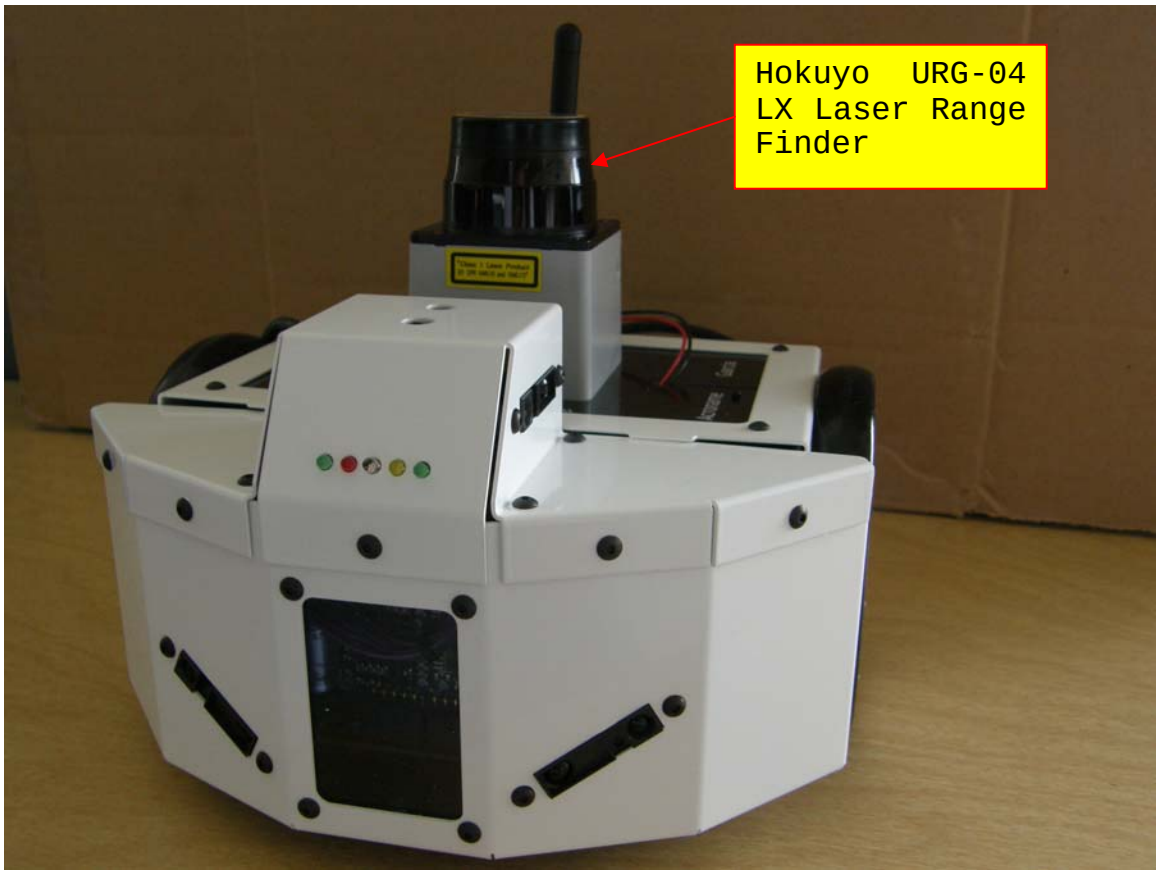


Figure 9. The Garcia Robot with the Hokuyo URG-04 LX Laser Range Finder on Top.

3. The Head

The head, seen in Figure 10, is located in the top portion of the Garcia robot. The head is comprised of a voltage regulator, power switch, charging jack, two infrared sensors, four Light-Emitting-Diodes (LEDs), an infrared receiver and an infrared transmitter. The LEDs can be used as a semi-diagnostic tool because each diode represents the health of a link. Starting from left to right:

- The GP Heartbeat: A blinking green LED provides the user the knowledge that the GP 2.0 is communicating with the Garcia robot.
- The Power Status: A solid red LED represents that the robot is receiving power and is in the on position.
- User LED: A solid yellow light provides the user knowledge that the Garcia is under a user’s control.
- The Moto Heartbeat: A blinking green LED signifies that the Moto 1.0 is communicating with the Garcia robot.

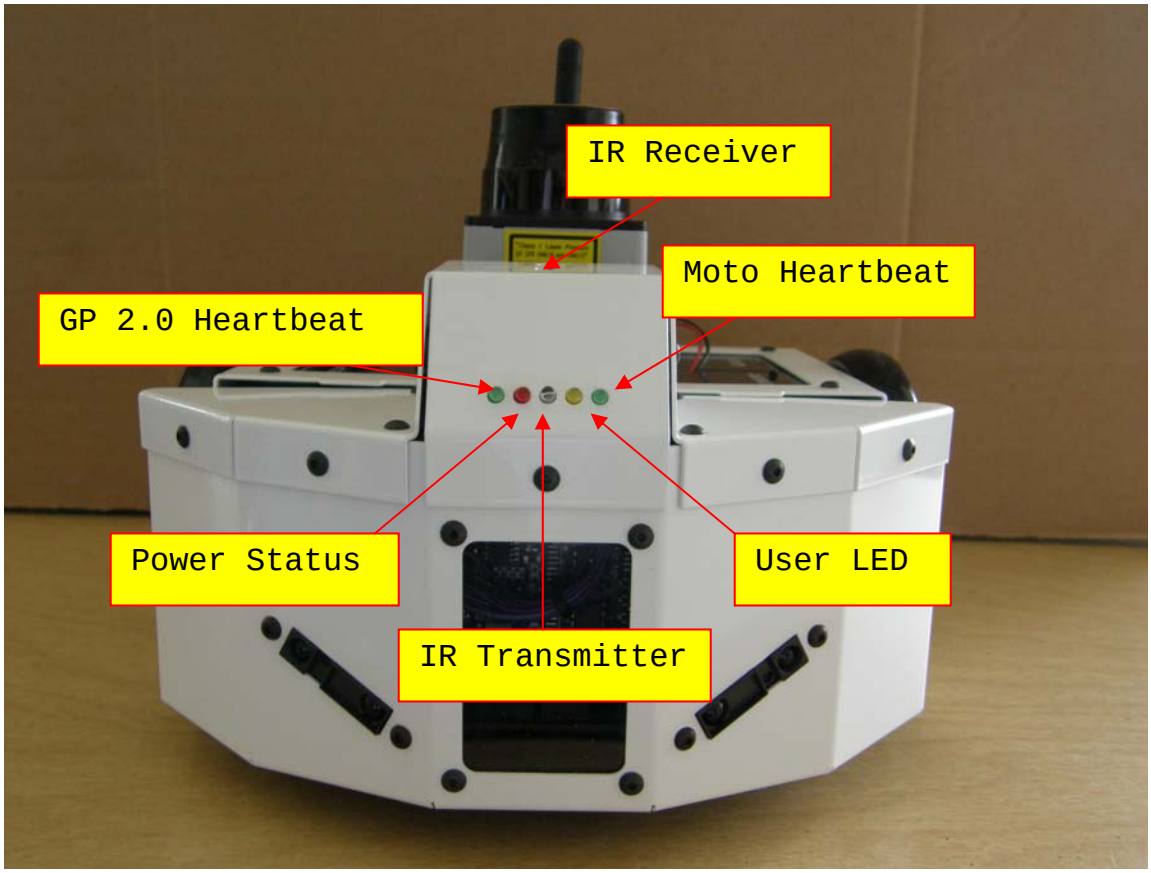


Figure 10. Front View of the Garcia Robot.

The infrared emitter allows the Garcia robot to communicate with other Garcia robots or handheld devices. The infrared receiver allows the Garcia to receive information from other Garcia robots or an infrared remote.

4. The Antenna

The Antenna used in this research is configured with a wireless radio frequency (RF) serial connection to the Gumstix Verdex Pro Motherboard in the Garcia robot.

C. HOST CONFIGURATIONS

The Garcia robot can operate in four configurations: no host, remote host, host on robot, and host network.

1. No Host

This configuration is the simplest to understand but the hardest to implement due to the fact that the programmer needs to have good knowledge of the Garcia's operating system. Additionally, this option uses the Garcia's controller as an operating system, drastically limiting computing power and storage.

2. Remote Host

In this option, the Garcia robot is connected, wirelessly or via serial cable, to a desktop computer. All of the functions are performed on the desktop computer and then sent to the Garcia robot for execution. Although this option allows for powerful computing, it is vulnerable to link delays.

3. Host on Robot

In this configuration, a handheld personal device (PDA) is carried onboard the Garcia robot.

4. Host Network

In this option, the Garcia robot carries a host computer onboard. Additionally, the robot has a wireless card that allows other computers to access the robot. When the robot is accessed by another computer, the host computer carried onboard the Garcia relinquishes the role of host

computer and behaves as a simple relay. The relay passes information from the new host, or remote computer, to the Garcia robot over the TCP/IP connection, seen in Figure 11.

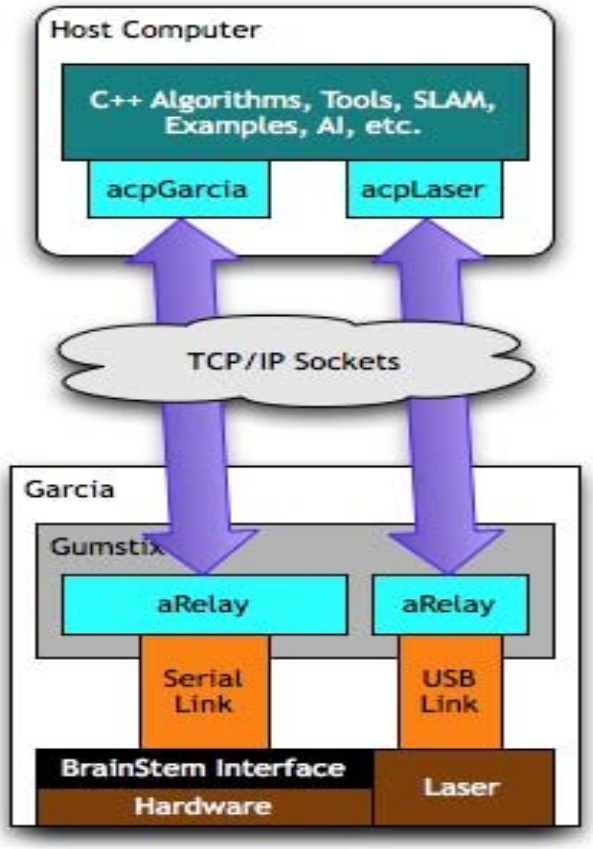


Figure 11. Pictorial Display of the Internal Workings of the Host Network Configuration [From 9].

In this research, the host network configuration was used because it allowed for the programmer to use a desktop computer without worrying about significant link delays. If the established link between the robot and the host computer fails, the robot will continue to monitor all of its sensors internally, therefore keeping the robot safe.

D. CHAPTER SUMMARY

This chapter provided an overview of the Garcia's internal hardware and external features. Also included in this chapter was a different host's configuration for the Garcia robot.

Chapter IV presents a laboratory related to simple movement.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. MOVEMENT LABORATORY

A. INTENTIONS FOR THE SIMPLE MANEUVERS LABORATORY

This research found the Garcia robot to be a very flexible tool that can be adapted to fit in any educational environment. Initially, operation of the Garcia robot can seem to be a daunting task to a new robotics student, but in reality the Garcia robot is a beginner-friendly platform. The intent of the following laboratory is to introduce students to simple commands that are used to control the Garcia robot. Additionally, this laboratory instructs students how to create and execute primitives, which are the building blocks of more complex operations. The laboratories were designed with the assumptions that the new students are familiar with the basics of C++ and Linux. If a student is not familiar with C++ or Linux, then the student would need to start with Appendix B, which provides an introduction to the Garcia robot's directories and Linux commands.

B. LABORATORY ONE: SIMPLE MANEUVERS

Purpose: To introduce the students to simple maneuvers such as moving forward and turning.

Procedure:

1. Log in to the computer.

2. Open terminal window and change directories to the aSource subdirectory, which is located under the acroname directory on the desktop. This can be done by typing the following on the command line: cd Desktop/acroname/aSource.

3. In the aSource directory, gain access to the aGarciaApp.cpp file using the gedit command (i.e, gedit aGarciaApp.cpp).

4. Once the text editor is open, a shell should appear that includes all the necessary libraries and header files. Also included in the shell are the Move and Turn classes and the main and run functions. For the purpose of this and subsequent laboratories, the user only needs to manipulate the run function. Additionally, the run function is where all the prescribed commands should be entered. In order to perform simple movements, the user will use the following commands to control the Garcia robot.

For the robot to move, the user must use the following commands, which calls on the Move class and provides an input of a pointer to the robot and a desired distance in meters:

```
pPrimitive = new Move(&Garcia, Distance);

garcia.queuePrimitive(pPrimitive);
```

For the robot to turn, the user must use the following commands, which calls on the Turn class and provides an input of a pointer to the robot, a turning radius which we have set to zero and a desired turn angle in radians:

```
pPrimitive = new Turn(&Garcia, 0.0f, angle_radians);

garcia.queuePrimitive(pPrimitive);
```

It is noted that the classes mentioned above use the wheelbase as a reference point, referring to Figure 12. In the Move class, if distance is a positive number

then the robot will move forward. Conversely, if the number is negative then the robot will move backwards. In the Turn class, if the turn angle is positive the robot will turn to the left. Conversely, if the angle is negative the robot will turn to the right.

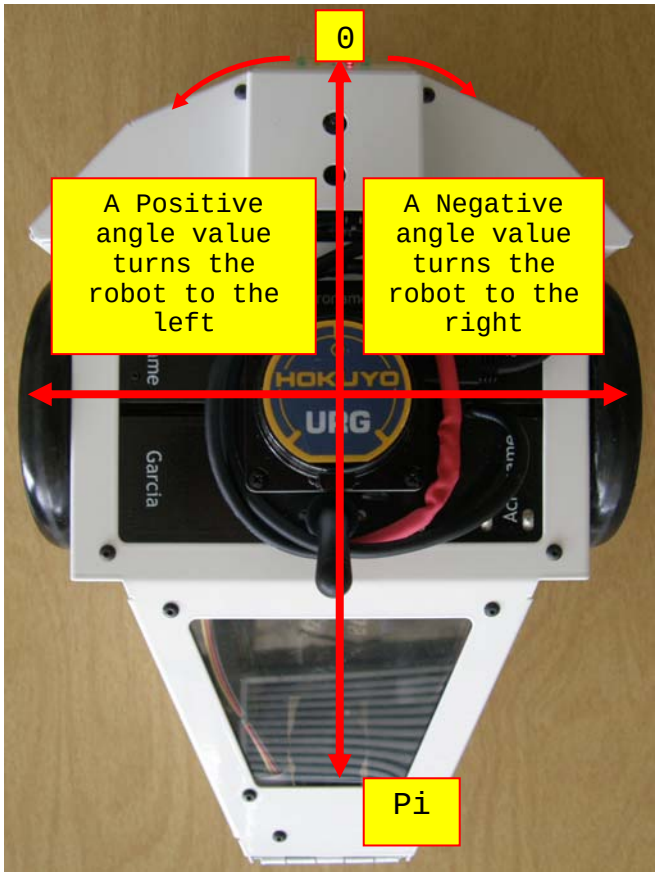


Figure 12. Robot References.

5. Modify the aGarciaApp.cpp file to make the Garcia robot perform a square.

6. After modifying the aGarciaApp.cpp file, the user must run the associated makefile, GarciaApp, to ensure that the changes have taken place. To do this, type the following in the command line: make –f GarciaApp.

7. If no errors occurred, go to the aBinary subdirectory and run the aGarciaApp executable.

8. Once the robot could perform a square, turn off the Garcia robot and log off.

Deliverables: Demonstrated the ability to make the Garcia robot perform a square of any size.

C. SOLUTIONS FOR LABORATORY ONE

An excerpt of code necessary to fulfill the deliverables is provided in Appendix C. The provided code commands the Garcia robot to make a square with sides equal to 0.5 meters. Additionally, the Garcia robot completes the square by making four left turns.

D. CHAPTER SUMMARY

This chapter introduced simple commands, such as Move and Turn, which control the Garcia robot movements.

Chapter V presents an introduction and use of IR sensors.

V. INFRARED LABORATORIES

A. INTENTIONS FOR THE INFRARED SENSORS LABORATORIES

In robotics, the use of sensors is paramount. Sensors are used for a myriad of tasks, ranging from determining distance to collision avoidance. The following laboratories expose students to the performance and implementation of IR sensors. During the exploration of IR sensor performance, the intent is to provide an insight to the accuracy of these sensors. Once an understanding of accuracy and performance has been reached, then IR sensors can be used as the “eyes” of the Garcia robot. The eyes can be used to identify obstacles and make decisions based on the location of the obstacle with respect to the robot. Furthermore, the IR sensors can be used to navigate a corridor, while keeping the Garcia robot safe from running into obstacles. IR sensors are introduced first, because they are easy to understand and implement. Additionally, students gain an appreciation of space awareness that is provided by the IR sensors.

B. LABORATORY TWO: GETTING ACQUAINTED WITH THE GARCIA’S INFRARED SENSORS

Purpose: To learn how to use IR sensors onboard the Garcia robotas well as to expose students to the limitations and performance of the IR sensors.

Procedure:

1. Log in to the computer.

2. Open terminal window and change directories to the aSource directory. The path is the following: Desktop/acroname/aSource.

3. In the aSource directory, gain access to the aGarciaApp.cpp file using the gedit command (i.e, gedit aGarciaApp.cpp).

4. In the aGarciaApp.cpp file, the user will use the following commands to enable and capture IR readings: garcia.getLeftFrontRanger(),
garcia.getRightFrontRanger(),
garcia.getLeftSideRanger(),
garcia.getRightSideRanger(), garcia.getLeftRearRanger() and garcia.getRightRearRanger().

It is noted that all of the measurements provided by the IR sensors are given with respect to the wheelbase. In this laboratory, the interest is placed on the collision, or “corrected,” distance. The “corrected” distance is the distance associated between the object and the robot. For example, if the front left IR sensor provides a reading, the reading must be corrected to reflect the actual distance from the front left panel on the robot to the object. Therefore, in order to get a corrected distance, the user must subtract the appropriate distance. The distances that must be subtracted are the following: .10 meters for the front and side sensor readings and .16 meters from the rear sensor readings. For example, if front left IR sensor reads .25 meters, then the object is actually .15 meters away from the robot. Additionally, a stand-off range of 8 centimeters, or 0.08 meters, has been

established for the front IR sensors. This stand-off range makes all the measurements of eight centimeters or less from the frame of the robot inaccurate. Also, if an object is past the detectable range of the IR sensors, the IR sensors will provide a reading of 0.00.

6. The user must use the previously mentioned commands to perform at least seven readings for each IR sensor. The user will position an object in front of a selected sensor and measure the distance to the object. The object should be placed at a distance between 0.08 to 0.4 meters from the location of the robot to ensure a valid IR sensor reading.

7. Save the changes and make the GarciaApp file.

8. After all required readings have been documented, turn off Garcia robot and log off.

Deliverables: The following table, Table 3, must be filled out. Additionally, Matlab graphs displaying actual distance versus "corrected" measured distance.

Front Left Range Sensor

Actual Distance (meters)	Measured Distance (meters)	"Corrected" Distance (meters)

Left Side Range Sensor

Actual Distance (meters)	Measured Distance (meters)	"Corrected" Distance (meters)

Rear Left Range Sensor

Actual Distance (meters)	Measured Distance (meters)	"Corrected" Distance (meters)

Right Left Range Sensor

Actual Distance (meters)	Measured Distance (meters)	"Corrected" Distance (meters)

Right Side Range Sensor

Actual Distance (meters)	Measured Distance (meters)	"Corrected" Distance (meters)

Rear Right Range Sensor

Actual Distance (meters)	Measured Distance (meters)	"Corrected" Distance (meters)

Table 3. Empty Sensor Readings Table.

C. SOLUTIONS FOR LABORATORY TWO

1. Tables

Table 4 displays all of the comparative data from the IR sensors.

Front Left Range Sensor

Actual Distance (meters)	Measured Distance (meters)	"Corrected" Distance (meters)
0.090	0.196	0.096
0.100	0.200	0.100
0.130	0.237	0.137
0.150	0.260	0.160
0.170	0.280	0.180
0.200	0.308	0.208
0.250	0.366	0.266
0.300	0.419	0.319
0.350	0.484	0.384

Left Side Range Sensor

Actual Distance (meters)	Measured Distance (meters)	"Corrected" Distance (meters)
0.030	0.124	0.024
0.050	0.159	0.059
0.090	0.193	0.093
0.100	0.201	0.101
0.130	0.239	0.139
0.150	0.259	0.159
0.170	0.281	0.181
0.200	0.304	0.204
0.250	0.383	0.283
0.270	0.401	0.301

Rear Left Range Sensor

Actual Distance (meters)	Measured Distance (meters)	"Corrected" Distance (meters)
0.040	0.204	0.044
0.050	0.215	0.055
0.090	0.258	0.098
0.100	0.267	0.107
0.130	0.301	0.141
0.150	0.321	0.161
0.170	0.341	0.181
0.200	0.385	0.225
0.250	0.447	0.287

Front Right Range Sensor

Actual Distance (meters)	Measured Distance (meters)	"Corrected" Distance (meters)
0.090	0.202	0.102
0.100	0.210	0.110
0.130	0.248	0.148
0.150	0.268	0.168
0.170	0.287	0.187
0.200	0.330	0.230
0.250	0.384	0.284
0.300	0.461	0.361
0.350	0.481	0.381

Right Side Range Sensor

Actual Distance (meters)	Measured Distance (meters)	"Corrected" Distance (meters)
0.030	0.128	0.028
0.050	0.157	0.057
0.090	0.196	0.096
0.100	0.204	0.104
0.130	0.247	0.147
0.150	0.269	0.169
0.170	0.292	0.192
0.200	0.336	0.236
0.250	0.391	0.291
0.270	0.401	0.301

Rear Right Range Sensor

Actual Distance (meters)	Measured Distance (meters)	"Corrected" Distance (meters)
0.040	0.210	0.050
0.050	0.218	0.058
0.090	0.266	0.106
0.100	0.282	0.122
0.130	0.312	0.152

0.150	0.330	0.170
0.170	0.370	0.210
0.200	0.424	0.264
0.250	0.443	0.283

Table 4. Comparative Data for the IR Sensors.

2. Graphical Display

Figures 13, 14 and 15 are associated graphical representation of the comparative data for the IR sensors:

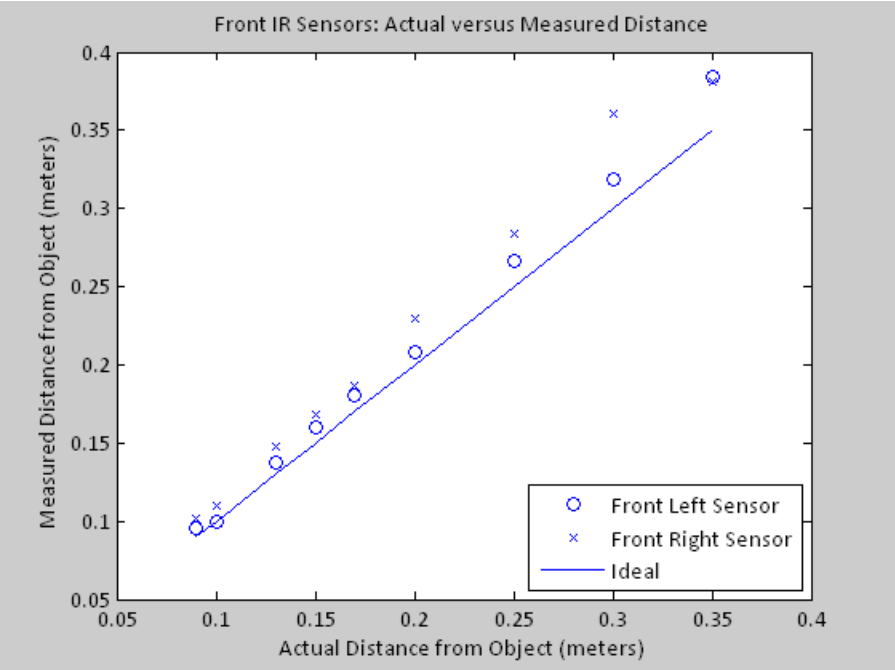


Figure 13. Front IR Sensor: Actual versus Measured.

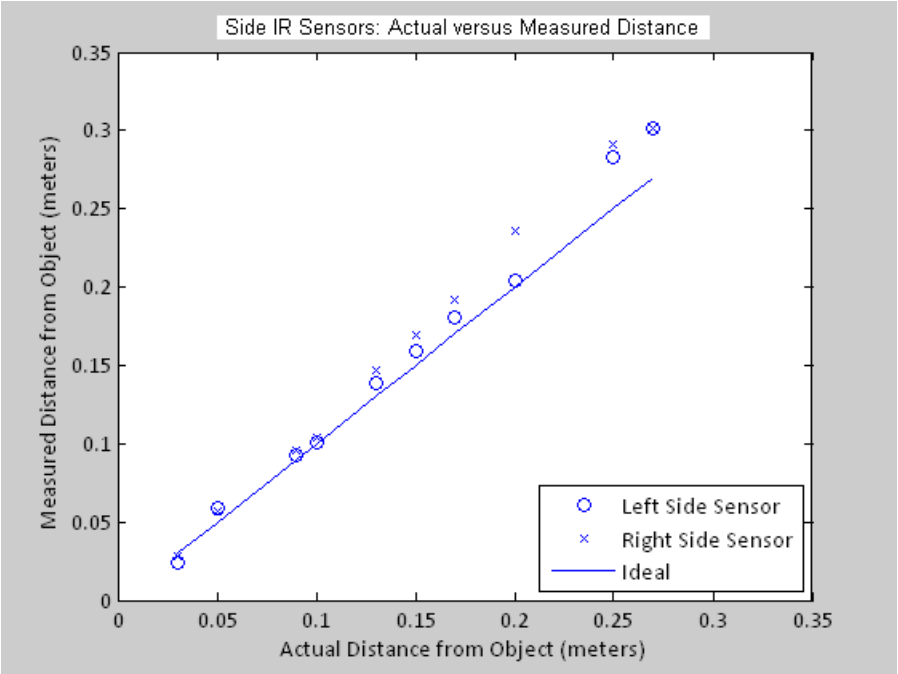


Figure 14. Side IR Sensor: Actual versus Measured.

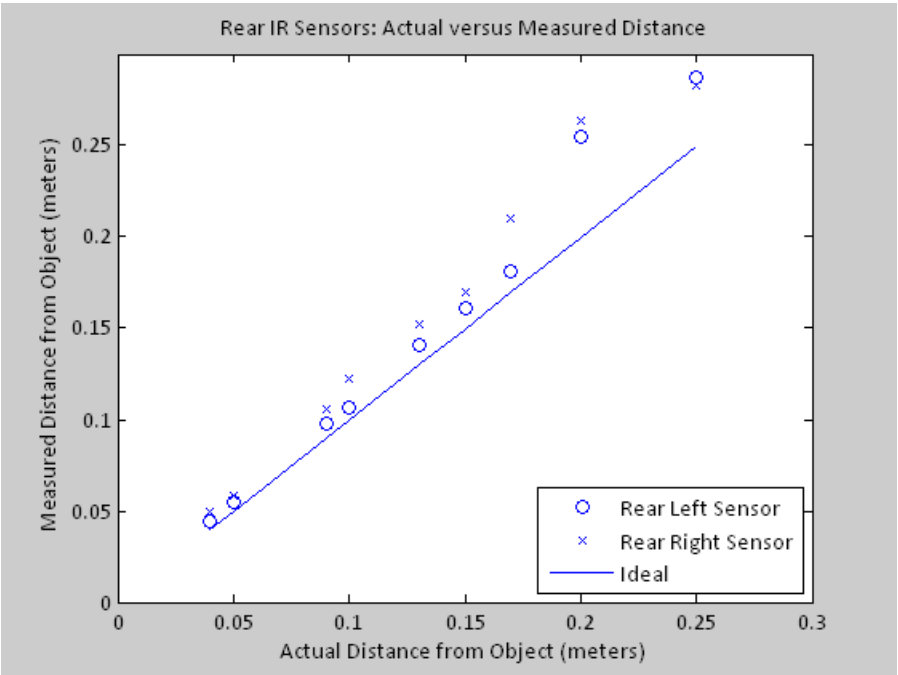


Figure 15. Rear IR Sensor: Actual versus Measured.

As expected, the measurement accuracy of the IR sensors decreases as the distance between the robot and the object

increases. Also of importance, the performance of the IR sensors differs from sensor to sensor.

3. Code

An excerpt of code necessary to fulfill the deliverables is provided in Appendix D.

D. LABORATORY THREE: OBSTACLE AVOIDANCE

Purpose: To learn how to use the IR sensors onboard the Garcia robot to perform obstacle avoidance.

Procedure:

1. Log in the computer.

2. Open terminal window and change directories to the aSource directory.

3. In the aSource directory, gain access to the aGarciaApp.cpp file.

4. In the aGarciaApp.cpp file, the user will use the commands learned in prior laboratories to develop a simple obstacle avoidance program using only the front IR sensors. The user must command the Garcia robot to travel a distance of six meters. While the robot is in motion, obstacles will be placed in its path. The Garcia robot should react to objects placed in its path. For example, if the Garcia robot “sees” an object on the right, then the Garcia robot must turn left to avoid collision. Additionally, if the robot “sees” an object on both left and right IR sensors, then it must go in the opposite direction. The goal is to have the

Garcia robot react to obstacle in its path as opposed to reaching the desired 6 meters.

Note: A pause of 2 seconds should be included to allow the Garcia robot to perform designated maneuvers. Otherwise, the Garcia robot will react to the same object multiple times.

5. Remember to make the GarciaApp as well as change directories to run the aGarciaApp executable.

6. Once the robot can react favorably to the obstacles in its path, turn off the Garcia robot and log off.

Deliverables: Demonstrated the ability to make the Garcia robot avoid obstacles in its path.

E. LABORATORY THREE SOLUTIONS

An excerpt of code necessary to fulfill the deliverables is provided in Appendix E.

F. LABORATORY FOUR: CORRIDOR TRAVEL USING IR SENSORS

Purpose: To learn how to use the combination of front and side IR sensors in order to enable the Garcia robot to travel along a corridor.

Procedure:

1. Log in the computer.

2. Open terminal window and change directories to the aSource directory.

3. In the aSource directory, gain access to the aGarciaApp.cpp file.

4. In the aGarciaApp.cpp file, use the commands learned in prior laboratories to develop a program that will enable the Garcia robot to travel along a corridor of any shape using the front and side IR sensors.

5. Remember to make the GarciaApp as well as change directories to run the aGarciaApp executable.

6. Once robot is able to travel through a corridor, turn off the Garcia robot and log off.

Deliverables: Demonstrated the ability to make the Garcia robot to travel along a corridor of any shape.

G. LABORATORY FOUR SOLUTIONS

An excerpt of code necessary to fulfill the deliverables is provided in Appendix F.

H. CHAPTER SUMMARY

This chapter introduced and utilized the IR sensors, onboard the Garcia robot, for a myriad of tasks to include range sensing, obstacle avoidance, and corridor travel.

Chapter VI presents an introduction and use of laser range finder.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. LASER LABORATORIES

A. INTENTIONS FOR THE LASER LABORATORIES

In robotics, the use of multiple sensors is usually the norm. Therefore, the following laboratories expose students to the use of a laser range finder for mapping the Garcia robot's surroundings. For simplicity, the laser range finder applications will not be used in conjunction with the IR sensors. The desire is to be able to map the Garcia robot's surroundings, while the robot is a stationary or mobile platform. During the mobile mapping, the intent is to reinforce the knowledge of transformations. A discussion on transformations can be found in [12].

B. LABORATORY FIVE: STATIONARY MAPPING USING THE LASER

Purpose: To learn how to use the laser range finder to perform mapping of the environment with a stationary Garcia robot.

Background: The Hokuyo URG-04 LX laser range finder will be used in the remaining laboratories. The Hokuyo is a class one laser. A class one laser is considered safe, and no extra safety equipment is needed while operating under normal conditions. The Hokuyo provides a detection area of 240 degrees, seen in Figure 16, and an angular resolution of 0.36 degrees. Therefore, the laser provides a total of 681 readings for every scan. Additionally, the information acquired from the laser is in meters with a resolution of approximately two millimeters. Lastly, the aGarciaApp.cpp shell

initializes the laser range finder. Therefore, the students only need to establish a TCP/IP connection in order to communicate with the laser. How to establish this link will be discussed in the procedure portion of this laboratory.

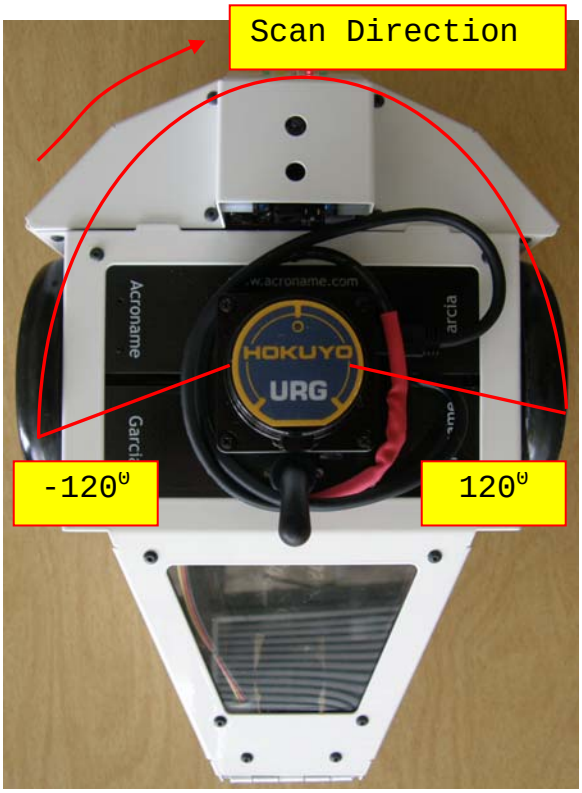


Figure 16. A Visual of the Hokuyo's Detection Area.

Procedure:

1. Log in the computer.

2. Open terminal window and change directories to the aSource directory.

3. In the aSource directory, gain access to the aGarciaApp.cpp file.

4. In the aGarciaApp.cpp file, the user will use the following commands to capture and log the desired

information from the laser range finder: laser.cmdScan(), laser.getReading(), and the if statement that follows:

```
if(logdata)
     fprintf(logfile_io,"\n");
```

The laser.cmdScan() commands the laser to perform one scan cycle, therefore, taking 681 readings. Then the user will need to utilize the laser.getReading() command in order to have access to the readings. Finally, everything associated with the if statement records the data to user designated file.

5. Place the Garcia close to a well defined feature, such as a wall or a corner—preferably a corner. Figure 17 displays an example of a well defined feature. Then use the command previously mentioned to interact with the Garcia robot’s laser. The user will need to command the laser to scan the surroundings. Then, the user will need to get and save the readings to a desired file.

Figure 17. The Garcia Robot Near a Well-defined Feature.

6. Prior to running the aGarciaApp executable, the user must establish a TCP/IP connection in order to communicate with the laser. The user will need to gain access to the processor on the Gumstix motherboard. Open a new terminal window. The secure shell (ssh) command will be used as follows to gain the desired access: ssh username@ip_address_destination. For example, if the user were to log in as root to the processor with an ip address of 192.168.2.110; the user would type the following in the command line: ssh root@192.168.2.110. Then the user will be prompted for a password, which will be provided by the laboratory staff. Now that the user has access to the processor, the user must change directories to the aBinary directory. Then, the user must execute the aRelay

executable from the command line. The aRelay can be executed by typing the following in the command line: ./aRelay –portname ttyACM0 (or ttyACM1) –port desired port number, usually above 8000. Now, the user is able to run the aGarciaApp executable.

7. Go back to the original window, make the GarciaApp file as well as change directories to run the aGarciaApp executable.

8. When running the aGarciaApp executable, the user must provide a desired file name to print the information gathered by the laser to that file. Executing the aGarciaApp and designating the desired file can be done by typing the following in the command line: ./aGarciaApp desired_filename.

8. Send the file that has the information gathered from the laser to a computer that has access to MATLAB in order to facilitate the plotting of the acquired information.

9. Once the laser information has been gathered, turn off the Garcia robot and log off.

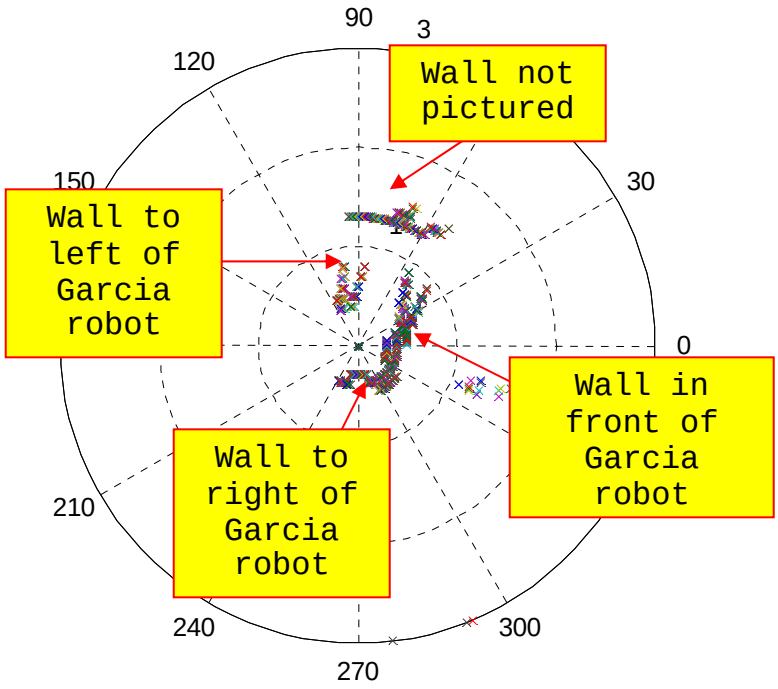
Deliverables: A map of the environment in Polar or Cartesian coordinate.

C. LABORATORY FIVE SOLUTIONS

1. Map of the Environment

At the start of this laboratory, the user was mandated to use a well established land mark. Therefore, the scene in Figure 17 was used to display the ability to map a desired

environment. By knowing the direction of which the scan occurs, the readings from the Hokuyo laser were coupled with their respective angle. The angles were designated as follows: The front of the robot was designated as the zero heading. Any angle falling to the left of the zero heading was considered negative. Conversely, any angle falling to the right of zero heading was considered positive. Figure 18 provides an accurate representation of the environment in Figure 17. Additionally, three prominent features, or walls, can be seen.



Note: The robot is located at the origin. Additionally, the front of the robot is zero heading

Figure 18. Map of the Environment.

2. Code

An excerpt of code necessary to fulfill the deliverables is provided in Appendix G. Additionally, the MATLAB code used to produce Figure 18 is provided in Appendix H.

D. LABORATORY SIX: MOBILE MAPPING USING THE LASER

Purpose: To learn how to use of the laser range finder to perform mapping of the environment on a mobile Garcia robot.

Procedure:

1. Log in the computer.

2. Open terminal window and change directories to the aSource directory.

3. In the aSource directory, gain access to the aGarciaApp.cpp file.

4. In the aGarciaApp.cpp file, use the commands learned in prior laboratories to develop a program that will enable the Garcia robot to travel through a pre-defined route, as well as map the surroundings. Figure 19 displays the corridor in which the Garcia robot needs to travel and map, simultaneously. The Garcia robot will need to take multiple readings while traveling through the corridor in order to map the whole corridor.

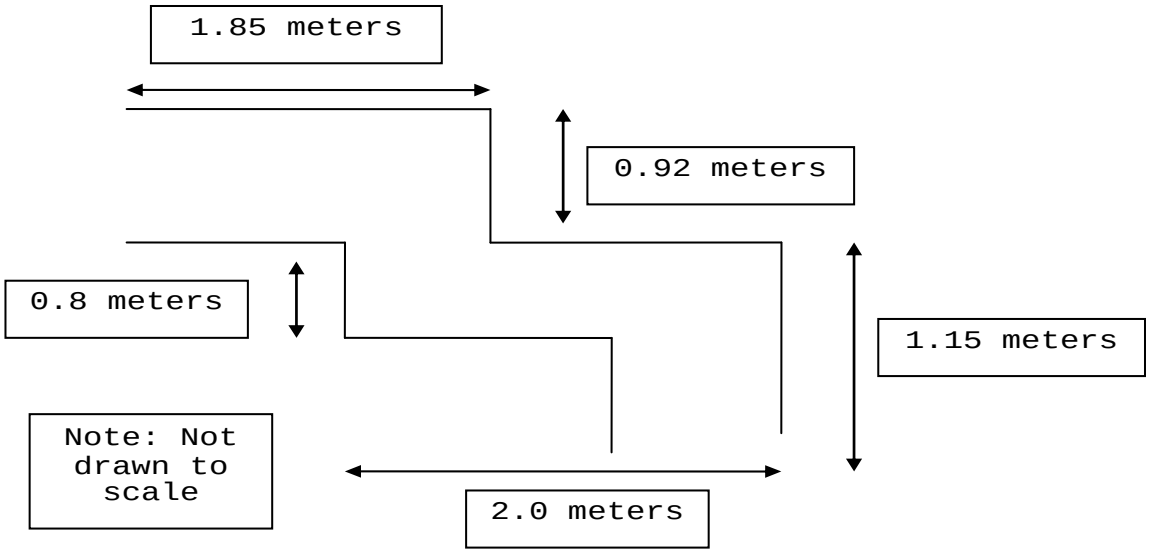


Figure 19. Corridor Used for Mobile Mapping Using the Garcia Robot.

5. Remember to establish a TCP/IP connection to allow communication with the laser. Additionally, remember to designate the file name of where the laser information will be logged.

6. Remember to make the GarciaApp as well as change directories to run the aGarciaApp executable.

7. Once the robot is able to travel through the corridor and the information from the laser has been collected. Send the file that has the information gathered from the laser to a computer that has access to MATLAB in order to facilitate the plotting of the acquired information.

8. Once the laser information has been gathered, turn off the Garcia robot and log off.

Deliverables: An illustration displaying the locations of where each reading was taken. Additionally, an environment map corresponding to each location of where the readings were taken. Lastly, combine all the reading to provide a map of the total corridor. Hint: Remember to use transformation matrix to relate each reading back to the world coordinates.

E. LABORATORY SIX SOLUTIONS

1. Illustration of Corridor

While travelling through the corridor, a total of nine readings were taken. The dimensions of the corridor were provided; therefore, the robot was commanded to take a pre-planned route, as well as to scan the environment at specified locations. The Garcia robot was commanded to do the following:

1. Perform a laser scan of surroundings.
2. Move forward 0.5 meters.
3. Perform a laser scan of surroundings.
4. Move forward 0.4 meters.
5. Perform a laser scan of surroundings.
6. Make a 90 degree turn to the Left.
7. Perform a laser scan of surroundings.
8. Move forward 0.7 meters.
9. Perform a laser scan of surroundings.
10. Move forward 0.6 meters.
11. Perform a laser scan of surroundings.

12. Make a 90 degree turn to the right.

13. Perform a laser scan of surroundings.

14. Move forward 0.35 meters.

15. Perform a laser scan of surroundings.

16. Move forward 0.4 meters.

17. Perform a laser scan of surroundings.

A visual representation of the pre-planned route and laser scans can be seen in Figure 20.

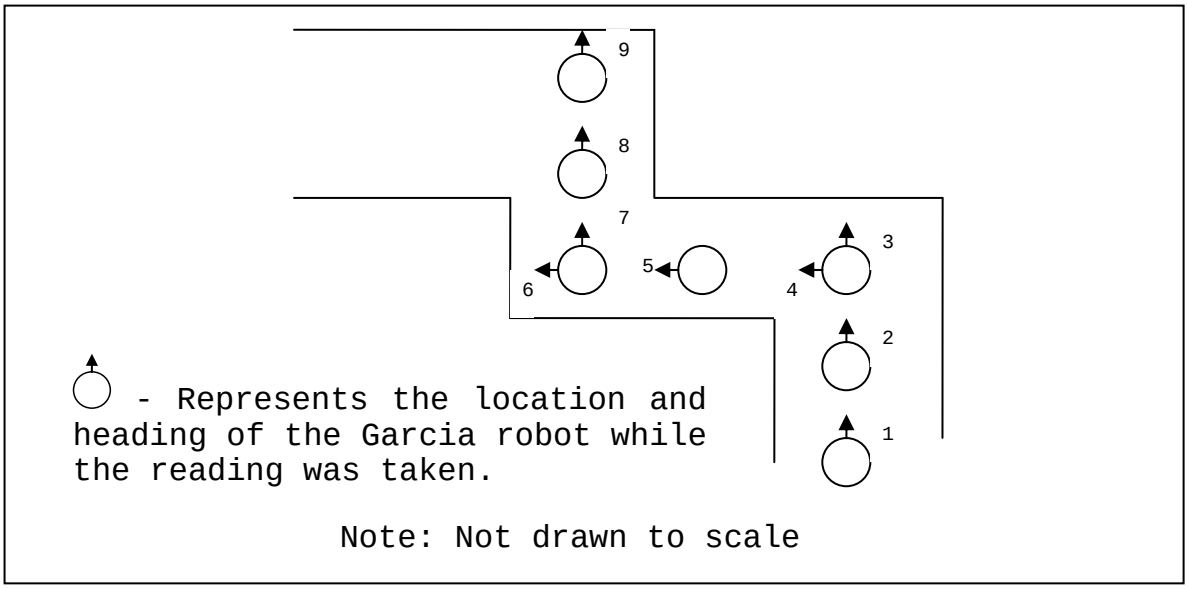


Figure 20. Visual Representation of Pre-Planned Route and Laser Scans Locations.

2. Map of the Environment

Figures 21 to 29 are maps of the environment corresponding to each location where a reading was taken.

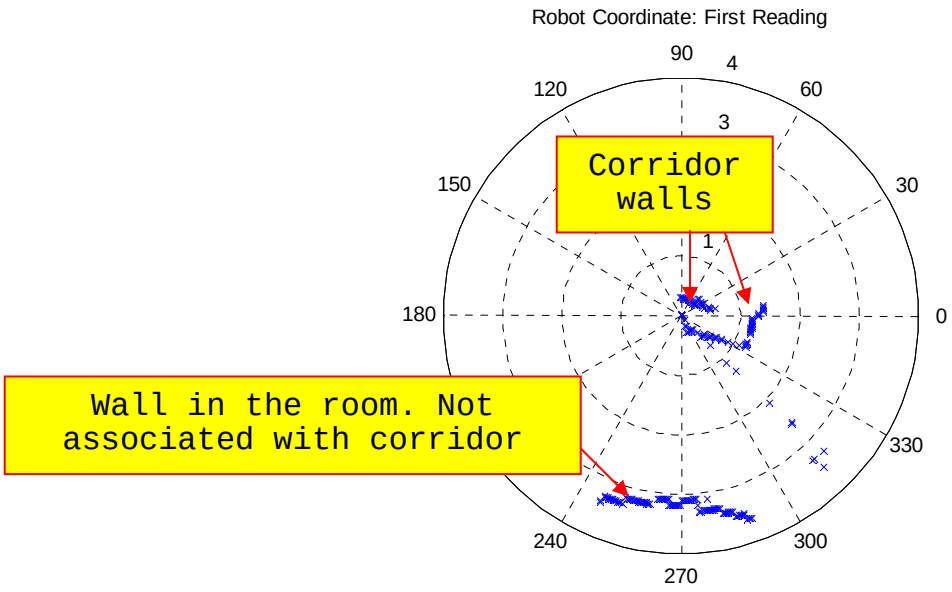


Figure 21. Map of the Environment Associated with the First Laser Reading.

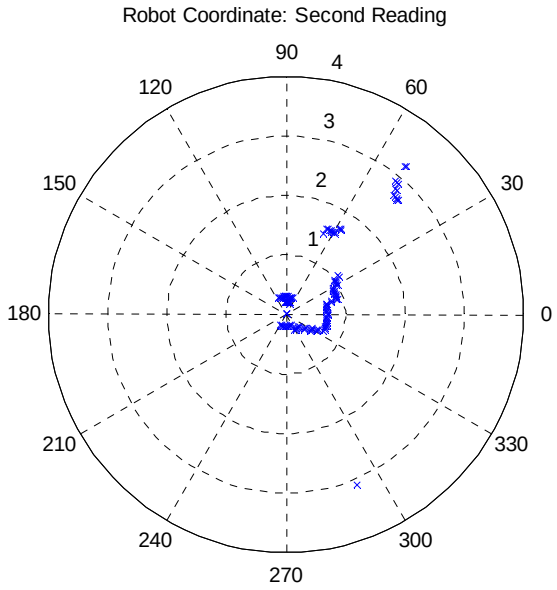


Figure 22. Map of the Environment Associated with the Second Laser Reading.

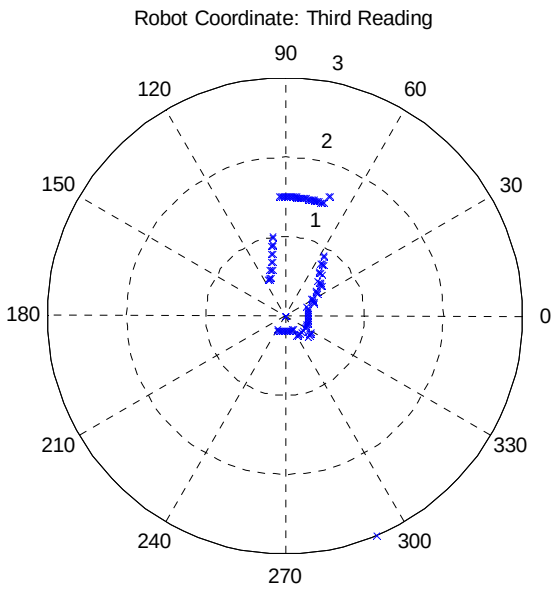


Figure 23. Map of the Environment Associated with the Third Laser Reading.

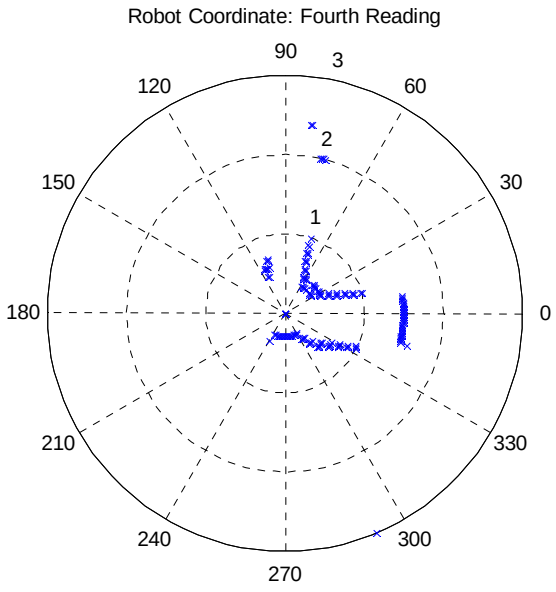


Figure 24. Map of the Environment Associated with the Fourth Laser Reading.

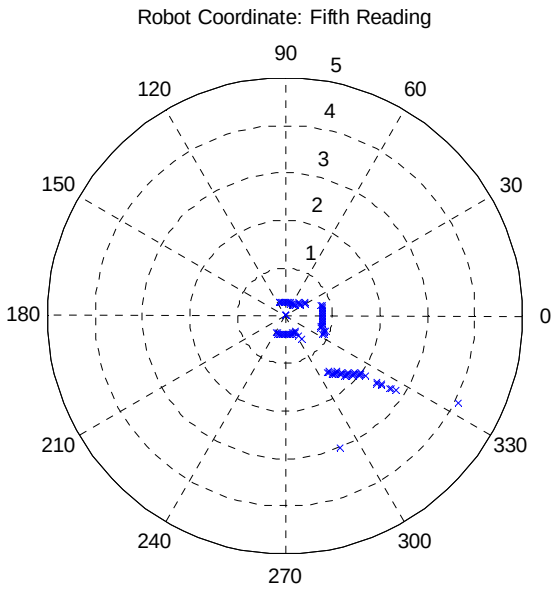


Figure 25. Map of the Environment Associated with the Fifth Laser Reading.

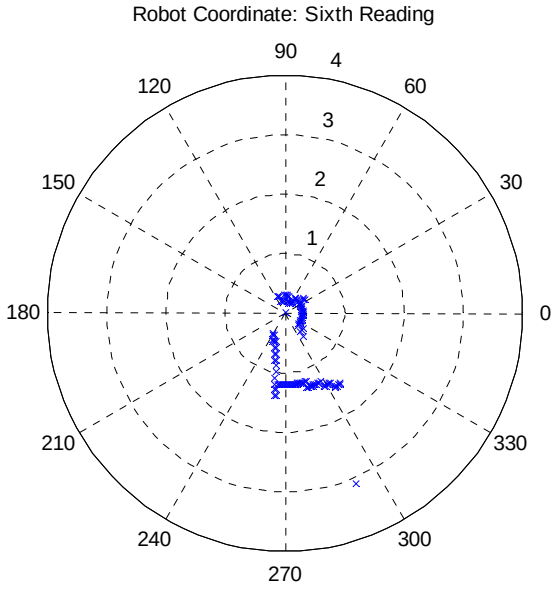


Figure 26. Map of the Environment Associated with the Sixth Laser Reading.

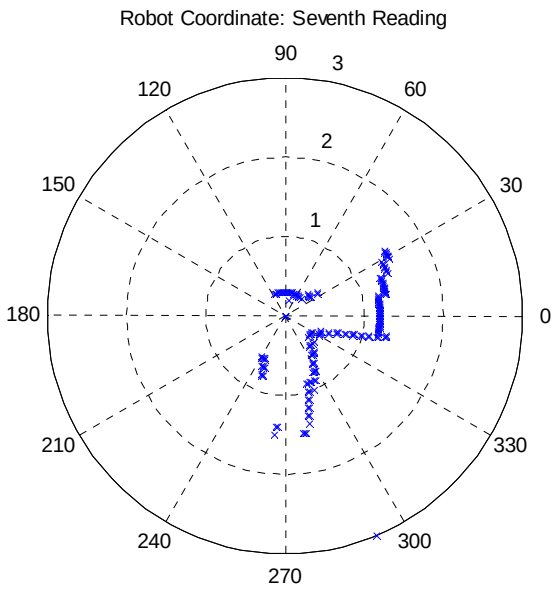


Figure 27. Map of the Environment Associated with the Seventh Laser Reading.

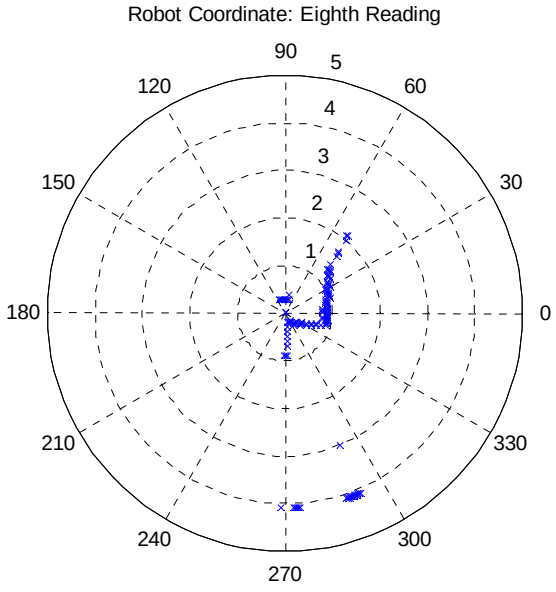


Figure 28. Map of the Environment Associated with the Eighth Laser Reading.

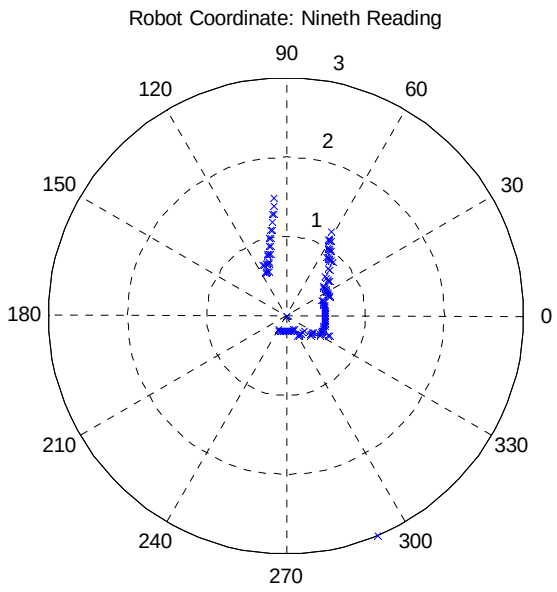


Figure 29. Map of the Environment Associated with the Nineth Laser Reading.

In order to attain an overall representation of the environment, each reading must be related back to a fixed coordinate system. In this case, the selected fixed coordinate system was coincided with the location and heading of the laser range finder during the first reading. The center of the laser range finder was considered to be the origin. Additionally, the zero heading of the laser range finder was considered to be the positive x-axis. Then, the positive y-axis was selected to be at the -90 degree heading of the laser range finder. Every subsequent laser reading was related back to this fixed coordinate by using the transformation matrix. The transformation matrix accounts for the translation and rotation of the robot with respect to the fixed coordinate system. Additionally, the two-dimensional space on which the robot operates was designated to be the x and y axes; therefore, rotations can

only happen about the z-axis further reducing the complexity of the transformation matrix. The general transformation matrix that was used in this experiment was the following:

$$T(\theta)=\begin{bmatrix}\cos(\theta) & -\sin(\theta) & 0 & X\\ \sin(\theta) & \cos(\theta) & 0 & Y\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix} \qquad 6.1$$

The X and Y values are simply the distance between the fixed coordinate origin and the center of laser range finder. The distances are with respect to the fixed coordinate system. The angle is determined by comparing the fixed zero heading to the current zero heading of laser. By associating each position and heading where a laser scan was performed, the laser reading could be used to attain a complete map of the environment. Therefore, for each time a scan was performed, the following computation was made:

$$^{o}P={}^{o}_{L}T*{}^{L}P \qquad 6.2$$

After relating all of the scan locations back to the fixed coordinate system, the information was plotted on Cartesian coordinate system to display a complete environment can be seen in Figure 30.

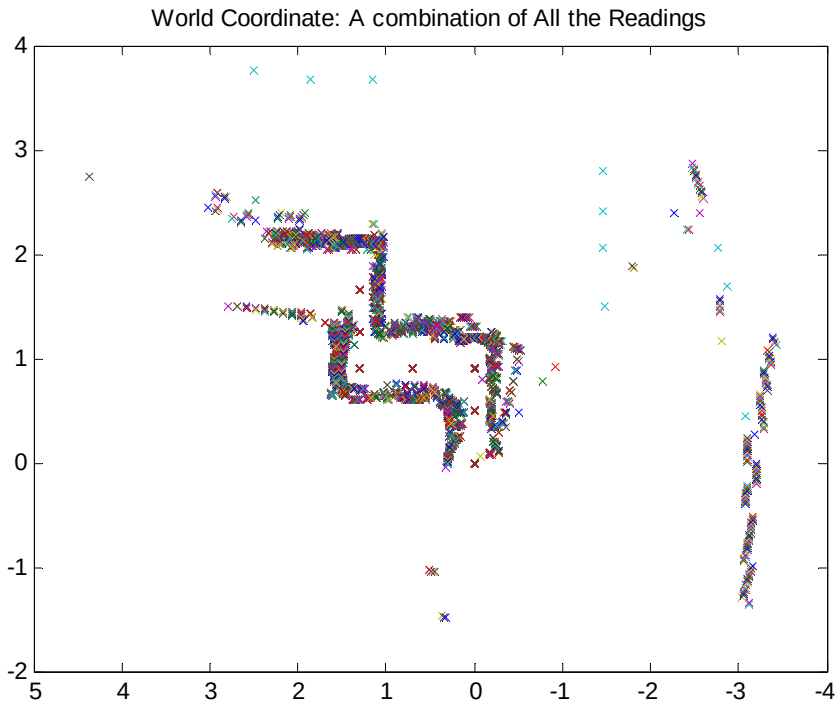


Figure 30. A Combination of All Nine Readings into a Cartesian Coordinate Plane.

3. Code

An excerpt of code necessary to fulfill the deliverables is provided in Appendix I. Additionally, the MATLAB code used to produce Figures 21 to 30 is provided in Appendix J.

F. CHAPTER SUMMARY

This chapter introduced and utilized the Hokuyo laser range finder for mapping purposes while the Garcia robot was stationary and mobile.

Chapter VII presents conclusions and future work.

THIS PAGE INTENTIONALLY LEFT BLANK

VII. CONCLUSION

A. SUMMARY

This thesis began with an overview of the research objective. Then, a comparison between three low-cost robots was performed using physical dimensions, onboard instrumentations and price as metrics. Ultimately, the Garcia robot was selected to investigate the feasibility of developing new academic laboratories. Next, an overview of the Garcia robot's internal hardware, external features and software interactions was presented. Subsequently, the thesis continued to the development of laboratories ranging from simple movement to environment mapping.

B. CONCLUSIONS

The investigation of the feasibility of developing laboratories on a low-cost commercially-available robot yielded mixed results. The positive results were that a low-cost robot, the Garcia, was found to be a flexible and powerful academic tool. The Garcia robot allowed for the development and implementation of the following laboratories: simple movement, investigation of IR sensor performance, obstacle avoidance and corridor travel using IR sensors, and mapping the environment using a laser range finder while the Garcia robot was stationary or mobile.

The drawbacks were that the Garcia robot was difficult to get started, because there was not a complete set of instructions for an initial set up. Additionally, the company was in transition between older and newer software.

Acroname allowed the use of their pre-release alpha software, which meant that some issues were not completely addressed. Some of the code provided by the company had errors, but the technical support engineers were quick to address the errors. Once past the initial issues, the Garcia robot was fairly easy to use. Unfortunately, the host configuration selected for this research limited the Garcia robot's performance. Due to the configuration, an initial latency of 15 to 20 seconds was observed, especially when communicating with the robot and the laser.

C. FUTURE RESEARCH

The Garcia robot is a flexible and powerful tool that provides many different avenues for research. There are multitudes of research topics with regards to robotics, but those of particular interest are listed below:

- Explore different host configurations on the Garcia to ensure maximum performance.
- Develop advanced laboratories for a robotics course.
- Conduct simultaneous localization and mapping (SLAM) research.

The Garcia offers a challenging and rewarding thesis. Future students should possess a strong C++ programming background with an emphasis on object-oriented programming.

APPENDIX A-INITIAL SET-UP/OUT-OF-BOX PROCEDURES

A. LABORATORY ESSENTIALS AND DOWNLOADS

This research used the following components to set up the laboratory environment:

- A base Garcia robot configured with a Gumstix Verdex Processor flashed with a Linux Operating System.
- Hokuyo URG-04 LX laser range finder.
- A desktop computer with a Linux Fedora 8.0 operating system.
- A wireless access point (WAP) from NetGear.
- A router from 3Comm.
- A USB to serial adapter.

In addition, there are two downloads, provided by Acroname Incorporated, that are required to get the Garcia robot functioning. The two downloads, which are zip files, are located at https://www.easierrobotics.com/cgi-bin/login. In order to download the files, the user must become a member of the site, which requires providing the user's name, email and password. After becoming a member and logging in, the user should see three different sections labeled: Blabs, Software, and Changes. Click on the "button" named Alpha Software under the Software section. Then download the file named Garcia ensuring that it is for a Linux platform. Also, download the file named C/C++ examples, again ensuring that it is for a Linux platform.

Both files should be downloaded to a desired location. For this example, the desktop was used. Now, the user has all the required components and files to operate the Garcia robot.

B. STARTING POINT

1. Set up a Wireless Network

Set up a wireless network by connecting the 3Comm router to an existing local network. Then connect the NetGear wireless access point to the 3Comm router. Now, gain access to the 3Comm router. Once access has been gained to the 3Comm router, click on LAN settings. Then inside the LAN setting tab, the unit configuration tab should automatically appear. Now, provide the desired values to the following fields: IP address, IP Pool Start Address, IP Pool Stop Address and enable the gateway to act as a DHCP server. Apply the changes to the settings. Then gain accesses to the NetGear wireless access point, under the IP settings tab provide the following values: an Access Point Name, enable the DHCP client, and enable the Spanning Tree Protocol. Then, under the wireless setting tab set the following values: SSID Broadcast to Enable, Channel/Frequency to 11/2.462GHz, Mode to g and b, and Data Rate to Best. Additionally, under the Security Setting tab, ensure that the security type is off. Also, under the Access Control tab ensure the Access Control is disabled. Apply all of the changes. After completing these actions, the user should have a functional wireless network.

2. Getting Access to the Desired Directories

Open a terminal window and go to the location of the downloaded files, the desktop for this case. The user should see the two files named Garcia_linux_i686.tgz and c_examples_linux_i686.tgz. Now, “unzip” the Garcia_linux_i686.tgz by typing the following command on the command line: tar –xzvf Garcia_linux_i686.tgz. This action creates the acroname directory with the following subdirectories and text files: aBinary, aInclude, aSource, aUser, aGarcia, aObject, aSystem, laser_readme.txt, relay_readme.txt, and console_readme.txt.

Again, go to the location of the downloaded files. “Unzip” the c_examples_linux_i686.tgz by typing the following command on the command line: tar –xzvf c_examples_linux_i686.tgz. This action created the examples directory with the following subdirectories: aGarciaApp, aGarciaKey, aIRLog, aLaserDemo. Now, the user has the desired directories and subdirectories.

3. Copying Desired Files from Examples Directory to the Acroname Directory

For the following explanations the username Tony will be used instead of providing a generic username. Again, the directories, acroname and examples, are on the desktop. From the desktop, go into the aSource subdirectory located under the acroname directory to start copying desired files. This is done by typing the following in the command line: cd acroname/aSource. The user will copy the following files from the aGarciaApp subdirectory under the examples

directory: aGarciaApp.cpp and makefile. To start copying the files type in the following commands in the command line:

cp /home/Tony/Desktop/examples/aGarciaApp/aGarciaApp.cpp .

This command will copy the aGarciaApp.cpp file from the aGarciaApp subdirectory under the examples directory to the aSource subdirectory under the acroname directory. Then type cp /home/Tony/Desktop/examples/aGarciaApp/makefile ./aGarciaApp. This command will copy the makefile from the aGarciaApp subdirectory under the examples directory and rename it aGarciaApp in the aSource subdirectory under the acroname directory. At this point, if the user types in ls in the command line; the user should see the makefile aGarciaApp and the aGarciaApp.cpp.

4. Configuring MiniCom

In a new terminal window, log in as root. Then type the following in the command line: minicom –s. This will allow the user access to the configure minicom. After typing minicom –s, the configuration menu will appear. Enter the values in Table 5 under the Serial Port Setup:

Serial Device	/dev/ttyUSB0 (edit to match your system as necessary)
Bps/Par/Bits	115200 8N1
Hardware Flow Control	No
Software Flow Control	No

Table 5. Important Values Under the Serial Port Setup Tab [After 10]

Then, return to the main configuration menu by hitting escape a couple of times. Then select Save Setup as dfl from the main configuration window. After the save is complete, select the Exit from Minicom. Minicom is now configured.

5. Communicating with the Gumstix Verdex Pro

Open the lid to the Garcia robot and located the FFUART connection on the Gumstix Verdex Pro motherboard. Connect the USB to serial adapter to the FFUART connection and the desktop computer. Again as root, type in following in the command line: minicom –o. This establishes a link of communication. The user should see a Welcome to minicom 2.2 in the top of the terminal window. Now turn on the Garcia robot and the user should see the universal bootloader (u-boot) in action. After the u-boot has completed loading, the Gumstix will require the user to provide a name and password, this is provided by the company. In this case, the name is root and the password is gumstix. Now, the user is communicating with the Gumstix Verdex Pro.

6. Changing the Name of the Garcia Robot (Optional)

At this point, type the command cd ../.. and then ls in the command windown. The user should see the following directories: bin, dev, home, media, proc, sys, usr, boot, etc, lib, mnt, sbin, tmp, and var. At this point, go into the etc directory by typing cd etc. The user should see a file named hostname. The user should call on the file by using the vi command, the Gumstix only has access to a vi editor. Type the following in the command line: vi hostname. This will open the hostname file and the name could be modified at this point.

7. Establishing a Wireless Connection

From the etc directory, go to the network subdirectory by typing the following in the command line: cd network and then ls. The user should see a file named interfaces. Open the file interfaces using the vi command. This will display all of the interfaces available to the Garcia robot. In the interfaces file, the user will only modify the portion named Wireless interfaces. Under the Wireless interfaces portion, add or modify existing code to the following:

- auto wlan0
- iface wlan0 inet dhcp
- wireless-mode manage
- wireless-essid NETGEAR

Remember to save the changes. Now type the following command in the command line to ensure that the changes take place: /etc/init.d/networking restart. To ensure that the changes have taken place, gain access to the 3Comm router and the name Garcia, or the name the user provided, should appear under the DHCP client tab. Now back out to the etc directory, modify the hosts file with the following: Change the original IP address given to the Garcia robot to the one provided by the 3Comm router. Again, type the following command in the command line to ensure that the changes take place: /etc/init.d/networking restart. Additionally, ping the 3Comm router, the NetGear WAC, and the Gumstix from the desktop computer to ensure proper communications links have been established. Now that the wireless connection has been established, the user can now use wireless features such as ssh and scp. The ssh, secure shell, command allows the user

to gain access to the Gumstix without a serial connection. Also of importance, the scp, secure copy, which allows the user the ability to securely copy files from a remote host. Both commands will be useful during and after the initial set up.

8. Receiving the Desired Files from the Desktop Computer to the Gumstix

Return to the initial terminal window, type the following in the command window: scp /home/Tony/Desktop/acroname/aGarcia/Garcia_linux_ARM.tgz root@192.168.2.110 (or the ip address that was given to the Gumstix by the 3Comm router) /home. The user should see a message stating the download is complete. Now go back to the terminal window that has connection to the Gumstix, back out to the home directory by typing the following in the command window: cd ../../../home. Then type ls to show all files in the directory. The user should see a “zipped” file named Garcia_linux_ARM.tgz. “Unzip” the Garcia_linux_ARM.tgz file by typing the following command in the command line: tar –xzvf Garcia_linux_ARM.tgz. The directory acroname should appear.

9. Preparing the Garcia Robot to go Wireless

The user should still have access to the Gumstix, back out to the main directory and enter the aBinary subdirectory in the acroname directory. The path to the aBinary is the following: /home/acroname/aBinary. At this point, the user will modify a file named relay.config. The user will modify the relay.config with the following values: port = 8001 or any number above 6000, portname = ttyS1 which points to the

Brainstem, and baudrate = 38400 which is the baud rate associated with the Brainstem. Save the changes. Then under the same directory, the user should see an executable named aRelay. The aRelay is an executable that forces the Gumstix to render the title of host computer to a remote computer. At this point, the Gumstix becomes a simple relay. To execute the aRelay, type the following in the command line: ./aRelay. The user should see two messages that read, relay resetting socket 192.168.2.110:8001, which is the IP address provided by the 3Comm router and the TCP/IP socket on which information is being received and relayed, on the next line, a relay statement with a blinking marker. If the above statements are true, the user has set up the relay.config correctly and the aRelay executable is performing correctly. At this point, type exit on the command prompt to stop the aRelay executable. Now the user is ready to make the aRelay executable run as part of the u-boot start up, for this portion please refer to [11] and follow the direction under Launching aRelay on Boot, but omitting cd from the last command. Therefore, the last command should read as follows: ln -s ../init.d/aStartup.sh S95aStartup. Additionally, the user must grant permission to this file by using the following command: chmod 777 aStartup.sh. Now, the user can disconnect the USB to serial adapter from the Gumstix motherboard on Garcia robot.

10. Ensuring the Wireless Communication is Working

Going back to the original terminal window, go to the acroname/aBinary directory. The user will need to provide the following permissions:

chcon –t textrel_shlib_t /home/Tony/Desktop/acroname/aBinary/libaStem.so,

chcon –t textrel_shlib_t /home/Tony/Desktop/acroname/aBinary/libaIO.so,

chcon –t textrel_shlib_t /home/Tony/Desktop/acroname/aBinary/libaUtil.so,

chcon –t textrel_shlib_t /home/Tony/Desktop/acroname/aBinary/libaMath.so.

After the permissions have been granted, it is time to ensure that the wireless communication has been set up correctly. The user will use the aConsole executable, but first the console.config file must be modified. The modifications should be set to the following: linktype to ip, ip_address = the same address that the Gumstix was assigned by the 3Comm router, ip_port to the same port number designated in the relay.config. Save the changes. Now run the aConsole executable by typing the following in the command line: ./aConsole. On the terminal window, you should see a message that reads, BrainStem Console Application. Additionally, the user will see a brainstem command line with a blinking marker in front. Also, the Garcia robot will have both green LEDs flashing or displaying a heartbeat. If both the Garcia robot and the terminal window are displaying the messages and the LEDs are flashing, then the wireless connection has been set up correctly.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B-LABORATORY ZERO: GETTING ACQUAINTED WITH THE GARCIA ROBOT'S DIRECTORIES AND LINUX COMMANDS

Purpose: To introduce students to the use of the computers in the Control Systems Laboratory. The computers have the Linux Fedora 8.0 as their operating system, which is used to control and interface with the Garcia robot, seen in Figure 31. In addition, students will learn to navigate between directories. A compilation of useful Linux commands are provided at the end of this laboratory in Table 6.

Figure 31. The Garcia Robot.

Background: The laboratory has been configured with a desktop computer that has a Linux Fedora 8.0 operating

system. Additionally, a wireless access point has been established through the use of a 3Comm router and a Netgear wireless access point. All of the components mentioned can be seen in Figure 32. The laboratory has been set up as a stand-alone wireless network, so the computers are isolated from the NPS network, but the computers are connected to the Internet.

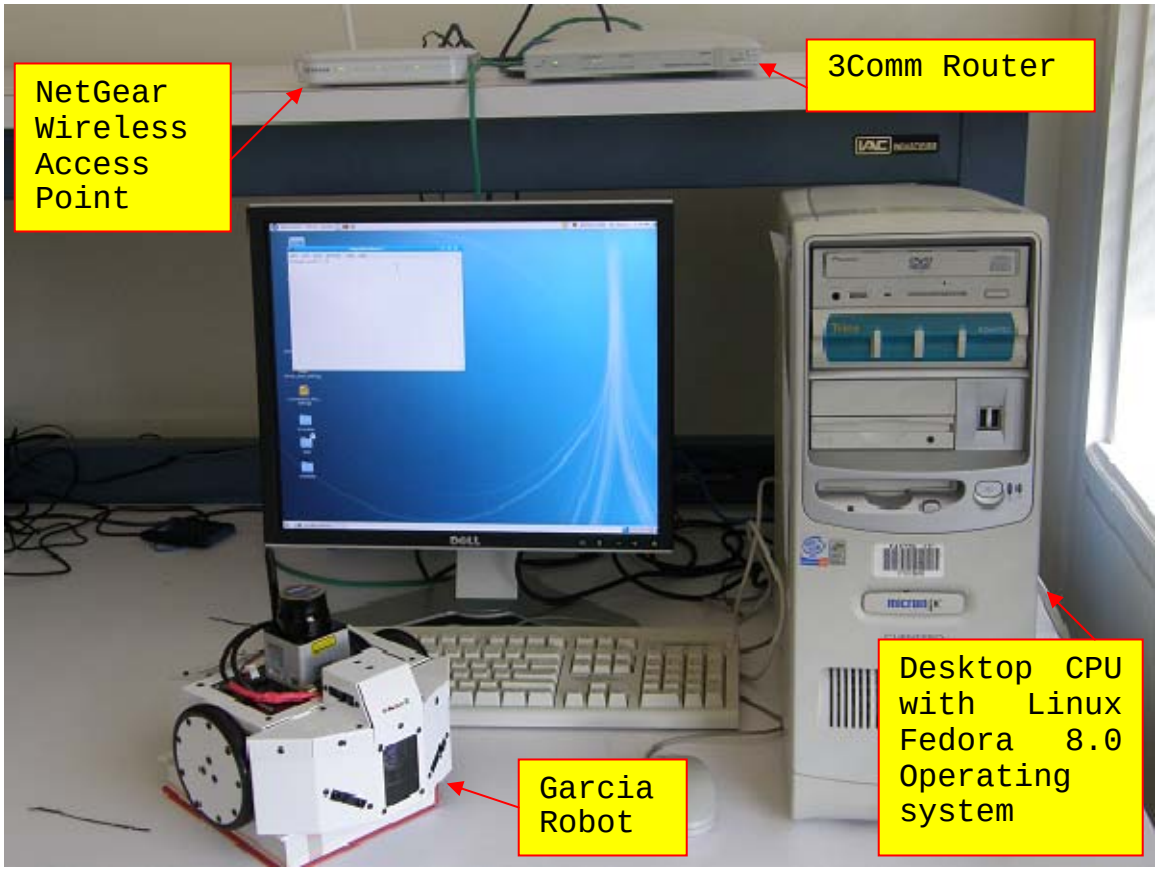


Figure 32. The Laboratory Set Up.

Procedures:

1. Log in to the workstation on which your account was created. The screen should appear as shown in Figure 33.

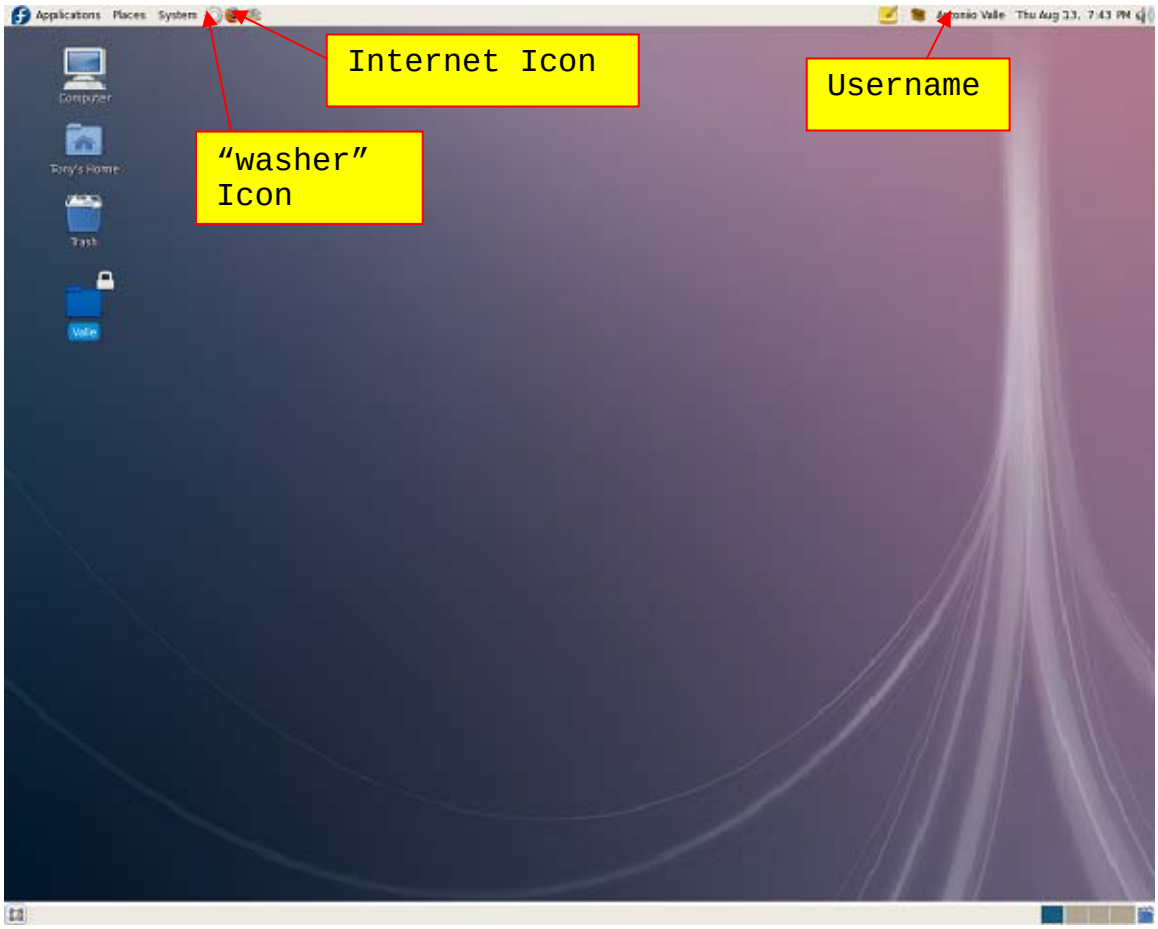


Figure 33. Screen Display After Logging in to Workstation.

After successfully logging in, open a terminal window. To open the command window, click on the icon that looks like a “washer” and select the last icon which is the terminal window, seen in Figure 34. A terminal window, seen in Figure 35, should open. The terminal window provides access to the command line that allows the user to navigate through a directory tree and execute and manage files.

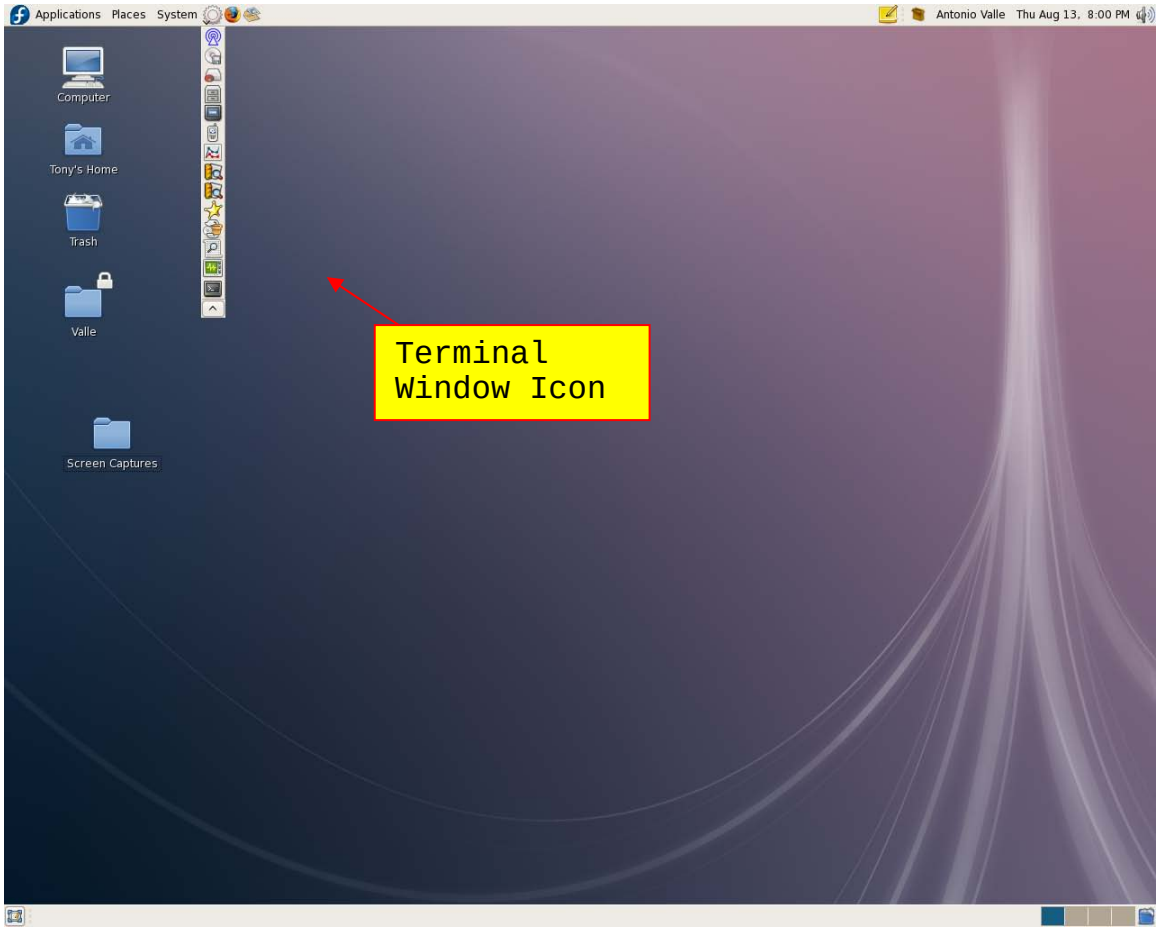


Figure 34. Display of Where to Find the Terminal Window.

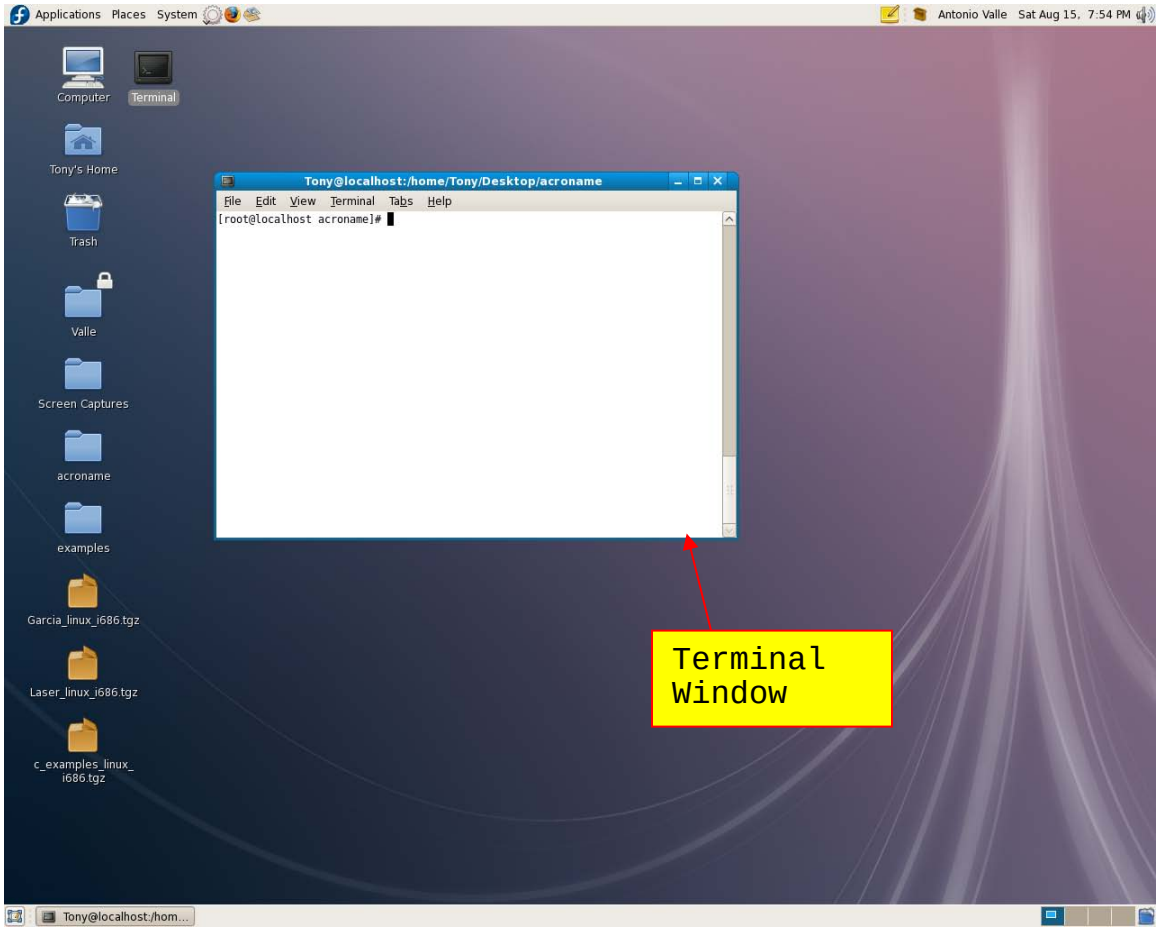


Figure 35. Display of Terminal Window.

2. Change into the directory that will be in used in the laboratory. Type the following in the command window: cd Desktop/acroname. The user is now in the acroname directory which includes the following subdirectories: aBinary, aInclude, aSource, aUser, aGarcia, aObject, and aSystem. This can be verified this by typing ls in the command window. A list of the previously mentioned subdirectories should appear.

3. The laboratories were designed such that users will only manipulate a handful of files in the aBinary and aSource subdirectories. At this time, go to the aSource by typing the following in the command window: cd aSource. Under the

aSource subdirectory, type ls in the command window. A list of files with the .cpp and .h extensions should appear. The acpGarcia.cpp and the acpGarcia.h are responsible for defining the Garcia class; therefore, these files will not be changed. The main file that will be manipulated will be the aGarciaApp.cpp, this file provides commands to the Garcia robot. To view and/or make modification to the aGarciaApp.cpp file type the following in the command line: gedit aGarciaApp.cpp. This command opens a text editor window which will have the aGarciaApp.cpp code. To close this window, simply click on the close icon, which is represented by the X. Another file of importance is the makefile named GarciaApp, because this makefile builds, or compiles, the environment. The GarciaApp is responsible for providing global definitions, pointing to sources files and flags, as well as, providing a location for the aGarciaApp the executable. Now, proceed to the aBinary subdirectory by typing the following in the command line: cd ../aBinary. Under the aBinary subdirectory, type ls in the command window. A list of executables should appear such as aConsole, aGarciaApp and aRelay, as well as, a few configuration files such as console.config, garcia.config, and laser.config. The executable files draw information from their respective configuration file. The configuration files been set to their desired inputs, so there is no need to manipulate the configuration files.

4. Turn on the robot allowing enough time for proper initialization. Then run the aConsole executable by typing the following in the command line: ./aConsole. Both green LEDs should begin to flicker displaying that the desktop

computer is communicating with the Garcia robot. Then exit the aConsole by typing “exit” in the command window.

5. The user will now learn to capture screen shots. Go to the applications tab, on the top left of the screen, and select accessories. Under accessories, select the option take screenshot. At this point, the user will need to make a decision of whether to capture the current window or the whole desktop. Then select take screenshot. A window appears that request a file name and location of where the screenshot will be saved. Additionally, a preview of the snapshot appears.

6. Turn off the Garcia robot and log off. By now, the user should be able to identify and view the files that will be manipulated, as well as navigate from directory to directory. Additionally, the user should be able to run executables and capture screenshots.

Deliverables: None.

Command	Description
cd ..	Moves up one level in the directories
cd "directory name"	Changes directory
clear	Clears screen
cp "source file" "destination file"	Copies the source file
ls	Shows directory contents
pwd	Shows path of current directory
gedit	text editor
vi	text editor
./"executables filename"	Performs executables in the current directory
make -f "filename"	Makes filename

Table 6. Compilation of Linux Commands.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C–LABORATORY ONE CODE

The following are excerpts of the aGarciaApp.cpp file that will make the Garcia robot perform a square with each leg being 0.5 meters:

```
for (int i = 0; i < 4;i++)
{
//movement one: moves 0.5 meters
pPrimitive = new Move(&garcia, 0.5f);
garcia.queuePrimitive(pPrimitive);


//turn 1: quarter turn
pPrimitive = new Turn(&garcia, 0.0f, aPI * 1/2);
garcia.queuePrimitive(pPrimitive);
}
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX D-LABORATORY TWO CODE

The following are excerpts of the aGarciaApp.cpp file that will make the Garcia robot read and display readings from the IR sensors:

```
////In the following section values are being read in from
the IR sensors. The value is being read ////a total of 5
times to ensure accuracy. The value is being displayed at
every instance.
     //Front
     for (int i=0; i < 5; i++)
     {
     FLsensor = garcia.getLeftFrontRanger();
     printf("\nFront Left= %6.3f\r",FLsensor);
     sleep(SLEEP_TIME);
     }
     for(int i=0; i <5; i++)
     {
     FRsensor = garcia.getRightFrontRanger();
     printf("\nFront Right= %6.3f\r",FRsensor);
     sleep(SLEEP_TIME);
     }

     //Side
     for (int i=0; i < 5; i++)
     {
     SLsensor = garcia.getLeftSideRanger();
     printf("\nLeft Side= %6.3f\r",SLsensor);
     sleep(SLEEP_TIME);
     }
     for(int i=0; i <5; i++)
     {
     SRsensor = garcia.getRightSideRanger();
     printf("\nRight Side= %6.3f\r",SRsensor);
     sleep(SLEEP_TIME);
     }

     //Back
     for (int i=0; i < 5; i++)
        {
     BLsensor = garcia.getLeftRearRanger();
     printf("\nBack Left= %6.3f\r",BLsensor);
```

```
sleep(SLEEP_TIME);
}

for(int i=0; i < 5; i++)
{
BRsensor = garcia.getRightRearRanger();
printf("\nBack Right= %6.3f\r",BRsensor);
sleep(SLEEP_TIME);
}
```

APPENDIX E–LABORATORY THREE CODE

The following are excerpts of the aGarciaApp.cpp file that will make the Garcia robot perform obstacle avoidance maneuvers:

```
while(1)
{
float FLsensor;
float FRsensor;

FLsensor = garcia.getLeftFrontRanger();
FRsensor = garcia.getRightFrontRanger();

if (FLsensor == 0 && FRsensor == 0)
     {
          printf("No obstacles\n");
          pPrimitive = new Move(&garcia, 300.0f);
          garcia.queuePrimitive(pPrimitive);
          sleep(SLEEP_TIME);
     }
     else if ((FRsensor == 0) && (.10f <= FLsensor))
     {
          printf("Obstacle on the left\n");
          pPrimitive = new Turn(&garcia, 0.0f, -aPI * 1/4);
          garcia.queuePrimitive(pPrimitive);
          sleep(SLEEP_TIME);
     }
     else if ((FLsensor == 0) && (.10f <= FRsensor))
     {
          printf("Obstacle on the right\n");
          pPrimitive = new Turn(&garcia, 0.0f, aPI * 1/4);
          garcia.queuePrimitive(pPrimitive);
          sleep(SLEEP_TIME);
     }
     else if ((.10f <= FLsensor) && (.10f <= FRsensor))
     {
          printf("Obstacle in front within turning radius.
          No way to get around. Turn around\n");
          pPrimitive = new Turn(&garcia, 0.0f, aPI * 1.0);
          garcia.queuePrimitive(pPrimitive);
          sleep(SLEEP_TIME);
     }
```

```
}
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX F–LABORATORY FOUR CODE

The following are excerpts of the aGarciaApp.cpp file that will enable the Garcia robot to travel along a corridor of any shape:

```
while (1)
{
     sleep(SLEEP_TIME);

     //reads in IR sensor readings and appropriate values
     //to L_sensor and R_sensor

     //reads in Front IR sensors
     FLsensor = garcia.getLeftFrontRanger();
     FRsensor = garcia.getRightFrontRanger();

     //reads in Side IR sensors
     SLsensor = garcia.getLeftSideRanger();
     SRsensor = garcia.getRightSideRanger();

///NOTE: IR sensors reads 0.000 when objects are greater
than approx .35 meters, so I am designating .40 meters
///for all IR values that read 0.000. This next section,
compares all the values to ensure the prior is
///implemented.
     printf("\nFront Left= %6.3f\r",FLsensor);
     if (FLsensor <= 0.000f)
     {FLsensor = .4f;}
     printf("\nFront Left= %6.3f\r",FLsensor);

     printf("\nFront Right= %6.3f\r",FRsensor);
     if (FRsensor <= 0.000f)
     {FRsensor = .4f;}
     printf("\nFront Right= %6.3f\r",FRsensor);

     printf("\nSide Left= %6.3f\r",SLsensor);
     if (SLsensor <= 0.000f)
     {SLsensor = .4f;}
     printf("\nSide Left= %6.3f\r",SLsensor);

     printf("\nSide Right= %6.3f\r",SRsensor);
     if (SRsensor == 0.000f)
```

```
     {SRsensor = .4f;}
     printf("\nSide Right= %6.3f\r",SRsensor);


//Adds up front and side sensors for each respective side
(i.e left and right). Therefore aiding
     Right = SRsensor + FRsensor;
     Left = SLsensor +FLsensor;

printf("\nTotal of Right Sensors = %6.3f",Right);


printf("\nTotal of Left Sensors = %6.3f",Left);

///Compares both totals to find the "white" space

     if (Right == Left)
     {
     printf("Even both sides. Going straight");
     pPrimitive = new Move(&garcia, 30.0f);
     garcia.queuePrimitive(pPrimitive);
     }

     else if (Right > Left)
     {
     printf("More white space on the Right. Going Right");
          if (0.5*SRsensor < FRsensor)
          {
          pPrimitive = new Turn(&garcia, 0.0f, -aPI * 1/8);
          garcia.queuePrimitive(pPrimitive);
          pPrimitive = new Move(&garcia, 0.15f);
          garcia.queuePrimitive(pPrimitive);
          }
          else
          {
          pPrimitive = new Turn(&garcia, 0.0f, -aPI * 3/8);
          garcia.queuePrimitive(pPrimitive);
          pPrimitive = new Move(&garcia, 0.10f);
          garcia.queuePrimitive(pPrimitive);
          }
     }
     else
     {
     printf("More white space on the Left. Going Left");
          if (0.5*SLsensor < FLsensor)
          {
```

```
            pPrimitive = new Turn(&garcia, 0.0f, aPI * 1/8);
            garcia.queuePrimitive(pPrimitive);
            pPrimitive = new Move(&garcia, 0.15f);
            garcia.queuePrimitive(pPrimitive);
            }
            else
            {
            pPrimitive = new Turn(&garcia, 0.0f, aPI * 3/8);
            garcia.queuePrimitive(pPrimitive);
            pPrimitive = new Move(&garcia, 0.f);
            garcia.queuePrimitive(pPrimitive);
            }
      }

}
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX G–LABORATORY FIVE CODE

The following are excerpts of the aGarciaApp.cpp file that will make the Hokuyo laser scan and record the environment:

```
if (laser_good && laser.cmdScan())
 {
     printf("Reviewing laser data\n");
     for (int i = 0; i < laser.getMeasurementSteps(); i++)
     {
          printf("%2.1f,   %2i\t",laser.getReading(i),i);
               if(logdata)
     fprintf(logfile_io,"%2.1f\t",laser.getReading(i));
               } // End of for loop
               if(logdata)
     fprintf(logfile_io,"\n");
     cout<<" logdata-"<<logdata<<"\n";
 } // end of scan
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX H–MATLAB CODE ASSOCIATED WITH LABORATORY FIVE

```
%%Laser Laboratory

%%getting the angles

count = 0

for i = 1:682
   theta_deg(i) = -120 + count*0.352;
   count = count+1;
end


theta_deg = theta_deg(:,2:length(theta_deg));
theta_rad = theta_deg.*pi/180;
rho = load('log1.txt');
[W,L] = size(rho);


for i = 1:W
theta_rad(i,:) = theta_rad(1,:);
end



%converts from polar to cartesian coordinates
%robot coordinates with the exception of first set of
points
[X_r,Y_r] = pol2cart(theta_rad, rho);


figure(1)
polar(theta_rad,rho,'X')
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX I–LABORATORY SIX CODE

The following are excerpts of the aGarciaApp.cpp file that will make Garcia travel through a predetermined path, as well as, make the Hokuyo laser scan and record the environment:

```
//initial reading
if (laser_good && laser.cmdScan())
          {
              printf("Reviewing laser data\n");
              for (int i = 0; i <
laser.getMeasurementSteps(); i++)
              {
                   printf("%2.1f,   %2i
\t",laser.getReading(i),i);
                   if(logdata)

     fprintf(logfile_io,"%2.1f\t",laser.getReading(i));

              }

              if(logdata)
                   fprintf(logfile_io,"\n");
              cout<<" logdata-"<<logdata<<"\n";

               }

//First Move: Moves forward 0.5
              printf("/nMove forward 0.5 meters\n");


                              pPrimitive = new
Move(&garcia, 0.50f);

     garcia.queuePrimitive(pPrimitive);
                              sleep(SLEEP_TIME);


//second reading of first corridor
if (laser_good && laser.cmdScan())
          {
              printf("Reviewing laser data\n");
```

```
               for (int i = 0; i <
laser.getMeasurementSteps(); i++)
               {
                    printf("%2.1f,   %2i
\t",laser.getReading(i),i);
                    if(logdata)

     fprintf(logfile_io,"%2.1f\t",laser.getReading(i));

               } // End of for loop
               if(logdata)
                    fprintf(logfile_io,"\n");
               cout<<" logdata-"<<logdata<<"\n";

                } // end of scan

//Second Move: Moves Forward 0.4
               printf("Move forward 0.4 meters\n");
                              pPrimitive = new
Move(&garcia, 0.40f);

     garcia.queuePrimitive(pPrimitive);
                              sleep(SLEEP_TIME);
//third reading of first corridor
if (laser_good && laser.cmdScan())
          {
               printf("Reviewing laser data\n");
               for (int i = 0; i <
laser.getMeasurementSteps(); i++)
               {
                    printf("%2.1f,   %2i
\t",laser.getReading(i),i);
                    if(logdata)

     fprintf(logfile_io,"%2.1f\t",laser.getReading(i));

               } // End of for loop
               if(logdata)
                    fprintf(logfile_io,"\n");
               cout<<" logdata-"<<logdata<<"\n";

                } // end of scan


//Third Move: turns pi/2
               printf("Turns pi/2/n");
```

```
                              pPrimitive = new
Turn(&garcia, 0.0f, aPI * 0.5);

     garcia.queuePrimitive(pPrimitive);
                              sleep(SLEEP_TIME);
//Fourth reading ---first reading of second corridor if
(laser_good && laser.cmdScan())
           {
               printf("Reviewing laser data\n");
               for (int i = 0; i <
laser.getMeasurementSteps(); i++)
               {
                    printf("%2.1f,   %2i
\t",laser.getReading(i),i);
                    if(logdata)

     fprintf(logfile_io,"%2.1f\t",laser.getReading(i));

               } // End of for loop
               if(logdata)
                    fprintf(logfile_io,"\n");
               cout<<" logdata-"<<logdata<<"\n";

                } // end of scan

//Fourth Move: move 0.7 meters
               printf("Move forward 0.7 meters\n");


                              pPrimitive = new
Move(&garcia, 0.70f);

     garcia.queuePrimitive(pPrimitive);
                              sleep(SLEEP_TIME);
                         ///scan here///2nd part of
corridor


//fifth reading ---second reading of second corridor if
(laser_good && laser.cmdScan())
           {
               printf("Reviewing laser data\n");
               for (int i = 0; i <
laser.getMeasurementSteps(); i++)
               {
```

```
                    printf("%2.1f,   %2i
\t",laser.getReading(i),i);
                    if(logdata)

     fprintf(logfile_io,"%2.1f\t",laser.getReading(i));

               } // End of for loop
               if(logdata)
                    fprintf(logfile_io,"\n");
               cout<<" logdata-"<<logdata<<"\n";

                } // end of scan

//Fifth Move: move 0.6 meters
               printf("Move forward 0.6 meters\n");

                              pPrimitive = new
Move(&garcia, 0.60f);

     garcia.queuePrimitive(pPrimitive);
                              sleep(SLEEP_TIME);

//sixth reading --- third reading of second corridor if
(laser_good && laser.cmdScan())
          {
               printf("Reviewing laser data\n");
               for (int i = 0; i <
laser.getMeasurementSteps(); i++)
               {
                    printf("%2.1f,   %2i
\t",laser.getReading(i),i);
                    if(logdata)

     fprintf(logfile_io,"%2.1f\t",laser.getReading(i));

               } // End of for loop
               if(logdata)
                    fprintf(logfile_io,"\n");
               cout<<" logdata-"<<logdata<<"\n";

                } // end of scan

//Sixth Move: turn -pi/2
               printf("Turn -pi/2/n");
                               pPrimitive = new
Turn(&garcia, 0.0f, -aPI * 0.5);
```

```
     garcia.queuePrimitive(pPrimitive);
                              sleep(SLEEP_TIME);

//seventh reading --- first reading of third corridor if
(laser_good && laser.cmdScan())
           {
               printf("Reviewing laser data\n");
               for (int i = 0; i <
laser.getMeasurementSteps(); i++)
               {
                    printf("%2.1f,   %2i
\t",laser.getReading(i),i);
                    if(logdata)

     fprintf(logfile_io,"%2.1f\t",laser.getReading(i));

               } // End of for loop
               if(logdata)
                    fprintf(logfile_io,"\n");
               cout<<" logdata-"<<logdata<<"\n";

                } // end of scan

//Seventh Move: move .35
               printf("Move forward 0.35 meters\n");
                              pPrimitive = new
Move(&garcia, 0.35f);

     garcia.queuePrimitive(pPrimitive);
                              printf("move 1 complete:
C3");
                              sleep(SLEEP_TIME);

//eighth reading second reading of third corridor if
(laser_good && laser.cmdScan())
           {
               printf("Reviewing laser data\n");
               for (int i = 0; i <
laser.getMeasurementSteps(); i++)
               {
                    printf("%2.1f,   %2i
\t",laser.getReading(i),i);
                    if(logdata)

     fprintf(logfile_io,"%2.1f\t",laser.getReading(i));
```

```
                } // End of for loop
                if(logdata)
                     fprintf(logfile_io,"\n");
                cout<<" logdata-"<<logdata<<"\n";

                 } // end of scan

//eight move: move 0.4
                printf("Move forward 0.4 meters\n");
                                pPrimitive = new
Move(&garcia, 0.40f);

     garcia.queuePrimitive(pPrimitive);
                                printf("move 2 complete:
C3");
                                sleep(SLEEP_TIME);

//nineth reading ---- third reading of third corridor if
(laser_good && laser.cmdScan())
           {
                printf("Reviewing laser data\n");
                for (int i = 0; i <
laser.getMeasurementSteps(); i++)
                {
                     printf("%2.1f,   %2i
\t",laser.getReading(i),i);
                     if(logdata)

     fprintf(logfile_io,"%2.1f\t",laser.getReading(i));

                } // End of for loop
                if(logdata)
                     fprintf(logfile_io,"\n");
                cout<<" logdata-"<<logdata<<"\n";

                 } // end of scan

//nineth move: turn pi/2
                printf("Turn pi/2\n");


                                pPrimitive = new
Turn(&garcia, 0.0f, aPI * 0.5);

     garcia.queuePrimitive(pPrimitive);
```

APPENDIX J–MATLAB CODE ASSOCIATED WITH LABORATORY SIX

```
%%Laser Laboratory

%%getting the angles
count = 0

for i = 1:682
   theta_deg(i) = -120 + count*0.352;
   count = count+1;
end


theta_deg = theta_deg(:,2:length(theta_deg));
theta_rad = theta_deg.*pi/180;
rho = load('lab6_2.txt');  %%got two sets of values lab6_1
and lab6_2
[W,L] = size(rho);


for i = 1:W
theta_rad(i,:) = theta_rad(1,:);
end
 %%initialized the world coordinates
X_w = [zeros(size(rho))];
Y_w = [zeros(size(rho))];


%converts from polar to cartesian coordinates
%robot coordinates with the exception of first set of
points
[X_r,Y_r] = pol2cart(theta_rad, rho);
X_w(1,:) = X_r(1,:);
Y_w(1,:) = Y_r(1,:);

%rotational matrix - translational
%1 inches = 0.0254 meters

%Tranformation used to convert second reading to world
coordinate
%moved forward 0.5 in the x-direction. x = 0.5
Move1 = [0.5;0;0;1];
T1 = [eye(4)];
T1(:,4) = Move1;
```

```
%Tranformation used to convert third reading to world
coordinate
%moved forward an additional 0.4 in the x-direction for a
total of x = 0.9
Move2 = [0.9;0;0;1];
T2 = [eye(4)];
T2(:,4) = Move2;

%Tranformation used to convert fourth reading to world
coordinate
%rotated pi/2 about the z-axis
theta1 = pi/2;
T3 = [cos(theta1), -sin(theta1),0,.9;
sin(theta1),cos(theta1), 0 , 0;
      0,0,0,0;0,0,0,1];

%Tranformation used to convert fifth reading to world
coordinate
%moved 0.7 in the positive y-direction. y = 0.7
theta1 = pi/2;
T4 = [cos(theta1), -sin(theta1),0,.9;
sin(theta1),cos(theta1), 0 , 0.7;
      0,0,0,0;0,0,0,1];

%Tranformation used to convert sixth reading to world
coordinate
%moved an additional 0.6 in the positive y-direction to a
total of y = 1.3
theta1 = pi/2;
T5 = [cos(theta1), -sin(theta1),0,.9;
sin(theta1),cos(theta1), 0 , 1.3;
      0,0,0,0;0,0,0,1];

%Tranformation used to convert seventh reading to world
coordinate
%rotated -pi/2 about the z-axis. Brings robot back to
original orientation.
theta1 = 0;
T6 = [cos(theta1), -sin(theta1),0,.9;
sin(theta1),cos(theta1), 0 , 1.3;
      0,0,0,0;0,0,0,1];

%Tranformation used to convert eight reading to world
coordinate
%moved an additonal 0.35 in the x-direction. x = 1.25
theta1 = 0;
```

```
T7 = [cos(theta1), -sin(theta1),0,1.25;
sin(theta1),cos(theta1), 0 , 1.3;
      0,0,0,0;0,0,0,1];

%Tranformation used to convert nineth reading to world
coordinate
%moved an additonal 0.4 in the x-direction. x = 1.65
theta1 = 0;
T8 = [cos(theta1), -sin(theta1),0,1.65;
sin(theta1),cos(theta1), 0 , 1.3;
      0,0,0,0;0,0,0,1];



%second reading in world coordinates
for i=1:length(X_r)

    World = T1*[X_r(2,i);Y_r(2,i);0;1];
    X_w(2,i+1) = World(1);
    Y_w(2,i+1) = World(2);

end

%third reading in world coordinates
for i=1:length(X_r)

    World = T2*[X_r(3,i);Y_r(3,i);0;1];
    X_w(3,i+1) = World(1);
    Y_w(3,i+1) = World(2);

end

%fourth reading in world coordinates
for i=1:length(X_r)

    World = T3*[X_r(4,i);Y_r(4,i);0;1];
    X_w(4,i+1) = World(1);
    Y_w(4,i+1) = World(2);

end

%fifth reading in world coordinates
for i=1:length(X_r)

    World = T4*[X_r(5,i);Y_r(5,i);0;1];
    X_w(5,i+1) = World(1);
```

```
    Y_w(5,i+1) = World(2);

end

%sixth reading in world coordinates
for i=1:length(X_r)

    World = T5*[X_r(6,i);Y_r(6,i);0;1];
    X_w(6,i+1) = World(1);
    Y_w(6,i+1) = World(2);

end

%seventh reading in world coordinates
for i=1:length(X_r)

    World = T6*[X_r(7,i);Y_r(7,i);0;1];
    X_w(7,i+1) = World(1);
    Y_w(7,i+1) = World(2);

end

%eight reading in world coordinates
for i=1:length(X_r)

    World = T7*[X_r(8,i);Y_r(8,i);0;1];
    X_w(8,i+1) = World(1);
    Y_w(8,i+1) = World(2);

end

%nineth reading in world coordinates
for i=1:length(X_r)

    World = T8*[X_r(9,i);Y_r(9,i);0;1];
    X_w(9,i+1) = World(1);
    Y_w(9,i+1) = World(2);

end

figure(1)
polar(theta_rad(1,:),rho(1,:),'X')
title('Robot Coordinate: First Reading')

figure(2)
polar(theta_rad(2,:),rho(2,:),'X')
```

```
title('Robot Coordinate: Second Reading')

figure(3)
polar(theta_rad(3,:),rho(3,:),'X')
title('Robot Coordinate: Third Reading')

figure(4)
polar(theta_rad(4,:),rho(4,:),'X')
title('Robot Coordinate: Fourth Reading')

figure(5)
polar(theta_rad(5,:),rho(5,:),'X')
title('Robot Coordinate: Fifth Reading')

figure(6)
polar(theta_rad(6,:),rho(6,:),'X')
title('Robot Coordinate: Sixth Reading')

figure(7)
polar(theta_rad(7,:),rho(7,:),'X')
title('Robot Coordinate: Seventh Reading')

figure(8)
polar(theta_rad(8,:),rho(8,:),'X')
title('Robot Coordinate: Eighth Reading')

figure(9)
polar(theta_rad(9,:),rho(9,:),'X')
title('Robot Coordinate: Nineth Reading')

figure(10)
plot(X_w,Y_w, 'X')
title('World Coordinate: A combination of All the
Readings')
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] “There’s No Right Side Up,” military.discovery.com. [Online]. Available: http://military.discovery.com/technology/robots/light-ugv/dragon-runner.html. [Accessed: Jul 12, 2009].

[2] Videre Design LLC, “ERA-MOBI,” videredesign.com. [Online]. Available: http://www.videredesign.com/robots/era_mobi.htm. [Accessed: Jul 31, 2009].

[3] Mobile Robots Incorporated, “Pioneer: The World’s Favorite For Versatility, Reliability and Durability,” activrobots.com. [Online]. Available: http://www.activrobots.com/ROBOTS/p2dx.html. [Accessed: Jul 31, 2009].

[4] Acroname Incorporated, “Acroname Robotics,” acroname.com. [Online]. Available: http://acroname.com/robotics/parts/G0-GARCIA-CONFIG.html. [Accessed: Jul 31, 2009].

[5] Videre Design Limited Liability Corporation, “EA-MOBI Feature Comparison To MobilieRobots’ P3DX,” videredesign.com. [Online]. Available: http://www.videredesign.com/robots/robot_mr_comparison.htm. [Accessed: Aug 01, 2009].

[6] Acroname Incorporated, “Garcia Manual,” acroname.com. [Online]. Available: http://acroname.com/garcia/man/man.html. [Accessed: Aug 02, 2009].

[7] Acroname Incorporated, “Gumstix Verdex Pro Package,” acroname.com. [Online]. Available: http://acroname.com/robotics/parts/R321-VERDEX-PRO-PKG.html. [Accessed: Aug 04, 2009].

[8] Acroname Incorporated, “Hokuyo URG-04LX Laser,” acroname.com. [Online]. Available: http://www.acroname.com/robotics/parts/R283-HOKUYO-LASER1.html. [Accessed: Aug 05, 2009].

[9] Acroname Incorporated, “Garcia API,” acroname.com, Apr. 08, 2009. [Online]. Available: https://www.easierrobotics.com/cgi-bin/reviewer. [Accessed: Aug 05, 2009].

[10] Gumstix, “Setting Up A Serial Connection,” gumstix.com. [Online]. Available: http://www.gumstix.net/Software/view/Getting-started/Setting-up-a-serial-connection/111.html. [Accessed: Aug 06, 2009].

[11] Acroname Incorporated, “Garcia aRelay Configuration,” acroname.com. [Online]. Available: http://www.acroname.com/garcia/starting/aRelay.html. [Accessed: Aug 07, 2009].

[12] J. J. Craig, “Spatial Descriptions and Transformations,” in *Introduction To Robotics: Mechanics and Control*, 3rd ed. New Jersey: Pearson Prentice Hall Incorporated, 2005, pp. 19-34.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, VA

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, CA

3. Hobart R. Everett
   SPAWAR Systems Center San Diego
   San Diego, CA

4. Xiaoping Yun
   Naval Postgraduate School
   Monterey, CA

5. Alexander Julian
   Naval Postgraduate School
   Monterey, CA